Just for your
TRS-80® COLOR
TDP System 100
Dragon & MC-10

The **RAINBOW** ®

THE COLOR COMPUTER MONTHLY MAGAZINE

The RAINBOW

1/85

ENGINEERS

Vol. IV
NO. 6

A Timesharing Crisis:

Beginners Issue

Elementary Graphics

A Simple Text Processor

The Interplanetary Fruit Fly

And BASIC Tutorials

TRS-80 Touch Pad

Radio Shack
TRS-80
COLOR COMPUTER 2

GO STATE

RAINBOW

Plus

**Scroll
Control**

**Rainbow
Checkbook**

**Graphics
Quickies**

Disk Test

**And,
More Than
Two Dozen Product
Reviews**

Under The RAINBOW

18

32

42

COVER art© by Fred Crawford

The small cassette tape symbols beside features and regular columns indicate that the program listings with those articles are on this month's RAINBOW ON TAPE, ready to *CLOAD* and *RUN*. For full details, check our RAINBOW ON TAPE ad on Page 220.

NEXT MONTH: In direct response to our RAINBOWfest-Princeton survey, February will be our "Utilities Issue" because of the high interest in, and demand for, these programming helpers.

We'll also kick off RAINBOW's third annual Adventure contest in addition to our usual wide spectrum of articles and programs. Look for THE RAINBOW for more on the Color Computer than is available from any other source!

COLUMNS

RAINBOWTECH

DEPARTMENTS

PRODUCT REVIEWS

LETTERS TO THE RAINBOW

ARTS AND LETTERS

Envelope Of The Month

John and Meeling
Walkerton, Ontario

PRIDE AND PREJUDICE

Editor:

Whether you ever publish this in your magazine or not, I've held the pride I feel in your publication back far too long!

Unfortunately, I'm an old lady and cannot read the computer language and I deeply regret that. However, I read all the English parts I possibly can and love every minute of it.

I'm sure the staff is excellent and your Editor is very fortunate to have you. I couldn't be more prejudiced than I am to your Editor, because I am Lonnie's Mom.

Mildred C. Falk
Birmingham, AL

WHAT'S A MOTHER TO DO?

Editor:

Regarding your article in the October '84 issue of RAINBOW, "Everyone Wants More RAM . . . ," I am a new CoCo owner and a mother of four kids. If I want "my turn" it has to be when the kids are in school or in bed so I relate to your problem in terms of my own dilemma; not enough RAM in my days and not enough K's in my CoCo. The K problem I am going to deal with by having a 64K package and a double-disk drive added to my unit. As for getting my share of time with the unit, well, the kids will eventually leave home and I'll get more time on the CoCo then.

I am enjoying my computer, but for my money RAINBOW has really added something special to our family's computer experiences. Keep THE RAINBOW bright!

Kathy Fjeldsted
Lundar, Manitoba

MAKING A COMPROMISE

Editor:

I think THE RAINBOW is an excellent magazine and is well laid out. Keep up the good work.

In your article ["Building October's Rainbow," Page 16] on what to put in THE RAINBOW (re: long 64K programs vs. short programs and more dialogue), I have this suggestion. Yes, we would like to see good, long version programs, but how about a compromise? Split the long programs over a couple of monthly editions. I don't think we will mind much and it gives a rest in between listings on the key-in. It would be to your advantage, too — it gives you more pages for articles and maybe increased circulation. Those who buy off-the-shelf magazines will have to buy next month's or make requests for last month's if they only have the second half.

John Felker
St. Catharines, Ontario

BLUEPRINTS FOR THE BUILDER

Editor:

The computer desk plans that Lloyd Wing (October 1984, Page 7) and other computer buffs may be interested in are in *Mechanix Illustrated*, February 1984.

The plans offer a flexible arrangement with locked compartment and a slide-out tray holding the computer keyboard.

Although the desk is made out of red oak plywood in the article and is estimated to cost $180, a less expensive grade of material could be used with excellent results.

Thomas Remakel
Dubuque, IA

Editor:

I have been buying magazines for 29 years for the purpose of learning and improving my abilities. THE RAINBOW is by far the best I have read. Enough, too much flattery might cause you to byte your bits.

In the October 1984 issue was a request for the location of some home built

computer table plans. I recommend the August 1984 *Popular Science*. There are several excellent computer table projects — winners of the annual plywood panel construction contest — well worth looking at and not expensive to build.

As for the already built, everybody with furniture is featuring some kind of computer furniture — at all kinds of prices.

Keep up the outstanding work and on to the next RAINBOWfest!

Robert A. Clark
Mooretown, NJ

ON THE HUMOROUS SIDE . . .

Editor:

It appears that in the October issue, "Information Please" section of the "Letters To THE RAINBOW" you must have left out much of Lloyd E. Wing's request, i.e., "I would like an example of both a purchased and homemade product: allowing for a monitor, disk drive and printer."

". . . another printer (for color), Y cable, ROM pack, another disk drive, cassette recorder, diskette filebox, another diskette filebox, chart and paper storage, six or more desk drawers, cassette filebox, another cassette filebox, notepad, pencil holder, trash basket, 20 gallon trash can — rubber, it hurts less when you kick it (Note: a 9600 Baud printer requires 33 gallons or better), desk lamp, aspirin holder, at least a 200-socket spike protected extension cord (a dedicated utility company may be substituted), modem, RS-232 switcher, printer interface, telephone, holder for THE RAINBOW magazines, bookcase, coffeepot with cup (an A.A. meeting list may be substituted), CoCo programming will require a change of socks and underwear (for all night sessions), calendar, radio, intercom (to let the family know of success — the sound of foot striking rubber trash can will suffice for failure), in-basket — nothing ever leaves, clock, checkbook (filled and solvent) a jillion stamps, envelopes, helpfully a Bible, picture of loved ones (for later comparisons), portable potty (not required if touch typist), dictionary for Adventures and a place to store joysticks . . ."

THE RAINBOW is doing Mr. Wing a great disservice by failing to correct his letter to a more realistic request.

DaDa and CoCo Hatton
Cockeysville, MD

HINTS AND TIPS

Editor:

I own a CoCo 2 which I recently upgraded to 64K. I found out that the IBM PC uses the same 4164 RAM chips. I was able to buy nine 150ns chips for $39 plus tax at an Austin PC store. So, when thinking of upgrading, check to see if there are any bargains at your local PC store before you settle for 200ns chips.

I've received a tremendous amount of enjoyment in the three months that I've owned my CoCo. The very first program I typed in was a "Sample Program #5" from the *Extended BASIC Manual*. If you like graphics, but don't enjoy typing long

listings, you'll love this one. It is short and sweet. Be sure to make the following changes for different effects:

```
1) 3 POKE 65495,0
   5 PMODE 4,1
   10 PCLS
   15 SCREEN 1,1
   80 GOTO 20
```

```
2) delete lines 30,35,65,70
   replace Line 40 with:
   40 COLOR RND(4)-1,RND(4)-1
```

First do the changes in part 1 and *RUN*, then do the changes in part 2 and *RUN*.

Daniel Ortiz
Austin, TX

BACK TALK

Editor:

I have been copying listings from THE RAINBOW into my Color Computer, but found it quite tedious to keep glancing from the magazine to the keyboard to the CRT screen. The other night I was wishing I could persuade my wife to read the listings to me, when it suddenly occurred to me I could dictate the listings to a cassette tape, then play that back while typing in the program. With a little experimentation in speed, groupings of characters, etc., I found this a big improvement. I used a foot switch, which is available at Radio Shack, to control the recorder while typing the listings.

I hope this helps others as much as it has helped me.

Russell R. Yost
Phoenix, AZ

EDITING ERRORS

Editor:

I have discovered that it is easier to correct errors if you type *EDIT* and the line number (e.g., *EDIT 100*) then press 'X' and backspace to the error. If you run out of paper while printing something, you can press the SHIFT @ key which will pause it, change paper and then press any key to start printing.

Robert Bowlby
Spencer, WV

Editor:

To all you people who wound up with an Epson MX-80 printer and Radio Shack's Disk Color *Scripsit*, and found out all you could print was garbage: if you use the print spool feature of *Scripsit*, not only will the printer work, but you'll be able to work on something else while it's printing the spool file.

Thanks for one terrific magazine. you make the CoCo worthwhile.

Andrew R. Ilowit
Hollywood, FL

WITH A FLIP OF THE SWITCH

Editor:

I have recently purchased the 64K Disk version of *Coco-Accountant II* from Federal Hill Software and the RGS Micro Dual DOS Card from Software Support.

Coco-Accountant II is a very good program; in fact, I am using it to do the

P.T.A. records. It does everything Federal Hill Software states, and is menu-driven.

The only problem is that the 64K Disk version is not compatible with JDOS. It must be used with Disk BASIC. This presented a problem in the beginning, as I did not know why the program did not function properly; however, with Federal Hill's help and assistance, we found that it was due to my use of JDOS. This is why I purchased the Dual DOS Card and Switch.

The directions that came with the Dual DOS Card were easy to follow and the installation was quickly performed. The hardest part was to drive a hole for the switch.

With this Card I can change from JDOS to Disk BASIC just by the flip of the switch and I don't have to worry about future incompatibility with JDOS.

Harold M. Bloom
Princeton, NJ

'CASTING' LINES

Editor:

If you would like to play *Gone Fishing* (THE RAINBOW, Jan. 1984, Page 158) and not have your line disappear, make the following changes in your program:

```
220 GET (125,128)-(135,136),M,G
600 LINE (149,40)-(149,K),PSET
610 PUT (140,K+0)-(150,K+6),M,PSET
```

Ted Mayor
East Greenbush, NY

POSITIVE OR NEGATIVE

Editor:

In the September 1984 issue [Page 6], Burnie Whiddon from Orlando gave a method of finding the exclusive OR of two integers. His method works for integers from 1 to 32767, but there is a simpler method to do the same thing and it works from -32767 to +32767. Of course, you have to know how the negative numbers are stored. Try the following lines:

```
10 INPUT"NUMBERS TO
   XOR";A,B
20 PRINT (A OR B) - (A AND B)
```

These will produce the desired XOR.

Mike Moore
Nepean, Ontario

ALL IN THE FAMILY

Editor:

As the publisher and exclusive marketing agent for the genealogical database *Family*, I would like to thank Mr. Howard Lee Ball for his praise of the program *Family* in a review of another genealogy program [*Ancestors 2.0* November 1984, Page 218]. The review states that Mr. Knight cooperated with the writer of *Ancestors* which is a disk-based system. *Family* was originally a tape-based system only.

We would like to inform the readers that Petrocci Freelance Associates has converted

Mr. Knight's *Family* to disk so it is now available on both tape and disk. The new disk version is $19.95. We feel it is only fair to Mr. Knight to make sure that readers know that his own program, *Family* is available in either format.

Susan Petrocci
Petrocci Freelance Associates
Tucson, AZ

Editor:

I am writing in response to the request of Mr. Dieter Klose in the November 1984 "Letters To THE RAINBOW" column. Mr. Klose inquired about computers and graphics printers for use in the production of his large-scale animated cartoon. Your Editor's Note suggested our program, *The Animator*, might help. While we thank you for the mention, I believe Mr. Klose had something else in mind. To set the record straight, *The Animator* does not support output to a graphics printer. A special "commercial version" of *The Animator* is being used by several graphics and advertising companies (making transfers directly to videotape). For Mr. Klose's purposes, we would recommend either a full-scale mainframe system or traditional film animation.

Stan Osterbauer, President
Triad Pictures Corp.
Sequim, WA

INFORMATION PLEASE

Editor:

Can someone send me a copy of the *Real Talker* "English to Phenome Dictionary?" They are no longer available from Colorware. I would be happy to pay for the copies. Thanks. Write to me at Rt.1, Box 103, 56444.

Gary Hansen
Deerwood, MN

Editor:

Congratulations on a fine magazine. Each month, with about five magazines of interest in the electronics field coming to me, yours rates tops.

I am interested in a program to help locate the settings for a TVRO (television receive only) antenna.

It would allow me to enter my latitude and longitude, degrees, minutes and seconds. Then the location of the desired satellite in the Smith Belt and tell me the elevation and degrees from true north to set the antenna.

With all the interest in earth stations, privately owned, if the program is not available maybe someone would be interested in writing one. A program writer I am not. Write me at Box 471, 44815.

Carl E. Nay
Bettsville, OH

Editor:

It has been suggested to me that your magazine might be able to assist me in finding out if there is a magazine subscription which might pertain to the MC-10 Color Computer put out by Radio Shack.

I am particularly interested in finding out what cassettes, if any, are available for this computer and where a person might obtain the same.

Ann Lobb
Hay River, Northwest Territories

Editor's Note: Please refer to Radio Shack's 1985 Catalog, Page 167.

GOLD IN ENGLAND

Editor:

As a refugee following the demise of *Chromasette*, I can see I have been missing a great deal in not subscribing to THE RAINBOW before now!

Do you have many readers in England? Maybe we can get some together to try and get reasonably priced software over here. Importing software from the United States results in having to pay six percent of value in customs charges and then VAT (tax at 15 percent on top of the converted dollar rate and the customs charge). The choice of non-Radio Shack software over here is severely limited.

Fancy finding the 'pot of gold' at the end of a rainbow over 4000 miles away. Isn't science wonderful!

Contact me at 14 Upcroft Ave., Edgware, Middlesex HA8 9RB England.

Norman Allen
Middlesex, England

Color Power II ™
Gives Your Co Co the Power of

®

CP/M

Color Power II is the CP/M computer which plugs into your Color Computer and marries its powerful Z-80A microprocessor with the outstanding 6809E microprocessor to allow you to run thousands of business programs including WordStar,® dBase II,® and SuperCalc II.™

Color Power II includes the CP/M 2.2 operating system and generates a high quality 80 column by 24 line display on your 80 column monitor with upper and lower case letters.

Many are developing hardware and software for use with **Color Power II.**

Double Density Software provides ULTRA TERM +, a communications program, and DOUBLE DOS II, an 80 column generator for Co Co basic. Morton Bay Software provides DOUBLE DRIVER, a monitor driver.

PRICES:
```
COLOR POWER II . . . . . . . . . . . . . . $329.00
ULTRA TERM + . . . . . . . . . . . . . . $ 56.00
DOUBLE DOS II. . . . . . . . . . . . . . $ 40.00
DOUBLE DRIVER or MONO II . . $ 25.00
```

Call us or one of our dealers or send check, Visa or MasterCard number with expiration date. N.J. residents add 6% sales tax.

(CPU) Color Power Unlimited, Inc.
1260 Springfield Ave., P.O. Box 606-F, New Providence, N.J. 07974 **(201) 665-9646**

DEALERS

Double Density Software
620 Kings Row
Denton, Texas 76201
817-566-2004

Morton Bay Software
316 Castillo Street
Santa Barbara, CA 93101
805-962-3127

DSL Computer Products, Inc.
P.O. Box 1176
Dearborn, Michigan 48121
313-582-8930

Micro R.G.S. Inc.
759 Victoria Square
Montreal H2Y 2J
800-361-5155

CP M is a trademark of Digital Research, Inc. WordStar is a trademark of Micropro International Corp.
dBase II is a trademark of Ashton-Tate   SuperCalc II is a trademark of Sorcim Corporation.

PIRATE PENDENCY

Editor:

Please accept my compliments on the most valuable peripheral I buy for my Color Computer. The high point of my month is always the arrival of THE RAINBOW. I would also like to cast my vote in favor of an article or series on FORTH.

I would like to commend these suppliers: Prickly-Pear Software, Spectral Associates, Spectrum Projects, Tom Mix Software and Double Density Software.

Lastly, I would like to toss out a few comments about software piracy. My question is to the software vendors who deal exclusively, or almost so, in imitations of arcade games. What legal right do these companies have to copy arcade games without licensing them from the original manufacturers? I hear a lot of whining and screaming from these manufacturers about people stealing their software, but it seems to me a case of the pot calling the kettle black. I invite rebuttal from the manufacturers.

Richard P. Adams
Mesquite, TX

KUDOS

Editor:

This magazine definitely is the best CoCo magazine around! The programs are always new, creative and exciting. It makes me feel good when I read about new hardware or software because I know you really know what you're talking about.

Keep up the good work on your great magazine!

Todd Amodeo
Winthrop, MA

Editor:

Congratulations on the success of your magazine. I'm a subscriber of your colorful magazine. I live in Canada and I've been wondering why nobody thought of doing a RAINBOWfest here in Canada. Everybody is 'CoCo' about THE RAINBOW magazine here.

Serge Larecque
Montreal, Quebec

APPLAUSE, APPLAUSE!

Editor:

As the New Brunswick RAINBOWfest was excellent, the Princeton RAINBOWfest exceeded it by far. I must say that both shows were worthwhile.

I have many, many companies to thank. To start off, for the New Brunswick show, I must thank Mark Data Products for their excellent Super-Pro keyboard and Tom Mix software for their quick delivery of SR-71.

For the Princeton show there are many more companies to thank. First on the list is Computer Plus for their service and the great price on the Okidata Microline 92 printer. (Not to mention the Okidata company for the quality of the printer.) Next, I should mention Micro World II's friendliness and good price on *Sam Sleuth* by Computerware. (Anybody know where some more evidence is on level 2? Got them to the 'lot' but arresting them doesn't work!)

How about MichTron for their Mach II joysticks, for giving us a free *Mudpies* game, and for a good price on *Graphicom*. Yes, they do discount at RAINBOWfests! Thanks to Moreton Bay Laboratory for their *Bjork Blocks (Graphicom* and *Bjork Blocks* are perfect together).

I'd also like to commend Compukit Corporation. Their 64K upgrade for $38, including installation, was a bargain. Although we had some trouble with it, bringing the computer back the next day remedied the problem.

Finally, and most importantly, I'd like to thank THE RAINBOW for supporting the shows and especially the CoCo. You guys have done a great job and we wouldn't be here without you.

Thanks to all.

Eric W. Lund
Millington, NJ

Editor:

Just a line to let you know how much my buddies and I like your magazine, THE RAINBOW. You seem to have an endless supply of goodies. I always try to point a new CoCo owner in the right direction — right to RAINBOW, that is.

"Cooking With CoCo" by Colin J. Stearman is a good example. I have enjoyed every segment so far. He is a fantastic writer in my opinion.

You have too many great articles for anyone to mention them all. As for my CoCo getting lonesome, not as long as I can get a RAINBOW! But my fingers are getting a little stiff, so I also get RAINBOW ON TAPE.

Keep all those great articles, and great people working for us.

Harold Lame
Gallatin, TX

Editor:

On Sunday, September 30, 1984, the drawing for the soft sculpture doll was held at RAINBOWfest in Princeton, N.J. Drawing the tickets for the PJCCC was Willo Falk of THE RAINBOW magazine. Standing by as verifying official was Dan Downard of THE RAINBOW. The winner of the doll was Paul Eckhard, RR 1, Box 152, Palmerton, PA.

Penn-Jersey Color Computer Club
Easton, PA

I really can't think of a nicer time of the year — and the only thing that makes this time of the year bad is that it isn't really that time yet.

I guess I should explain. This is the January issue of THE RAINBOW, but it comes out in mid-December. However, in order for me to meet the deadlines that have been imposed, I have to write it in early to mid-November. So, while I am trying to project myself a month ahead — it is a little hard to do. I'll try.

Because this is the January issue, I think it is important to look at the year which has just ended (or is about to end) and the one that is coming up. First, a few observations about 1984:

George Orwell's society did not take us over during 1984, as neither I nor you really thought that it would. At the same time, we did see computers and machines doing more and more things for people than ever before.

The computer business, in particular, was a good and a bad one. It was good for those of us interested in the CoCo because tens of thousands of users were added to the ranks. At the same time, more and more people looked at the Color Computer for the first time and this has made it what may very well be the most popular low-cost computer going.

For some other people the computer business has not been particularly good. We keep hearing about "the shakeout" — and a lot of people "got shook out." Computer companies. Software houses. Magazines. I've written about all this before, so there is nothing new here. Just an end-of-year observation that Tandy Corporation/Radio Shack seems to be as strong or stronger than ever, while some other people seem to be weaker or non-existent.

Looking ahead, I see the trend continuing. The fact of the matter is that many of the companies which show "new computers" at big shows like COMDEX one year, are nowhere to be seen the next year. One case in point is close to home: Tano Corporation — which had a big display for its U.S. Dragon computer last year. This time around they are not even in the market.

Why don't a lot of these people succeed? That is one of the concerns I think needs to be answered in 1985. And, frankly, I think the answer is that they do not succeed because they don't have anything new that will really grab the market.

The same goes in software. And magazines. No one is going to make inroads into an established market unless they can build a better mousetrap. Someone once wrote a book entitled *To Catch A Mouse, Make A Noise Like A Cheese.* It was a good book and it makes a lot of sense. But, you have to have a better cheese than the guy who has been selling you a perfectly good wheel of cheddar for a couple of years, or why would anyone switch? Too many people in the computer market are expecting people to switch simply because they have arrived on the scene.

What we need in 1985 are new things. New computers. New software. And not just new software, but new concepts. New ideas. I remember when *VisiCalc* first came out — it was the first program that gave the businessperson something he could really use. Apple Computer Co. sold a lot of computers thanks to ads which showed *VisiCalc* running.

I hear that several people are working on new endeavors in a lot of areas. I hope the trend continues and makes it to the marketplace. New programs will make your CoCo an even better investment — you'll be able to do more. And, of even more significance, you'll be able to do things you did not consider it possible to do. That's truly significant.

I see something new coming from a hardware standpoint, too. Whatever comes from Radio Shack will, undoubtedly, be upwardly compatible. That is something I have written about before, and it is a major strong point for the Radio Shack and Tandy line of computers.

In short, I see 1985 as an exciting year! I think it is a year in which the CoCo Community will continue to grow and the support for our favorite computer

Telewriter-64 ™
the Color Computer Word Processor

- ■ **3 display formats: 51/64/85 columns × 24 lines**
- ■ **True lower case characters**
- ■ **User-friendly full-screen editor**
- ■ **Right justification**
- ■ **Easy hyphenation**
- ■ **Drives any printer**
- ■ **Embedded format and control codes**
- ■ **Runs in 16K, 32K, or 64K**
- ■ **Menu-driven disk and cassette I/O**
- ■ **No hardware modifications required**

THE ORIGINAL

Simply stated, Telewriter is the most powerful word processor you can buy for the TRS-80 Color Computer. The original Telewriter has received rave reviews in every major Color Computer and TRS-80 magazine, as well as enthusiastic praise from thousands of satisfied owners. And rightly so.

The standard Color Computer display of 32 characters by 16 lines without lower case is simply inadequate for serious word processing. The checkerboard letters and tiny lines give you no feel for how your writing looks or reads. Telewriter gives the Color Computer a 51 column by 24 line screen display with *true lower case characters*. So a Telewriter screen looks like a printed page, with a good chunk of text on screen at one time. In fact, more on screen text than you'd get with Apple II, Atari, TI, Vic or TRS-80 Model III.

On top of that, the sophisticated Telewriter full-screen editor is so simple to use, it makes writing fun. With single-letter mnemonic commands, and menu-driven I/O and formatting, Telewriter surpasses all others for user friendliness and pure power.

Telewriter's chain printing feature means that the size of your text is never limited by the amount of memory you have, and Telewriter's advanced cassette handler gives you a powerful word processor without the major additional cost of a disk.

...one of the best programs for the Color Computer I have seen...
— Color Computer News, Jan. 1982

TELEWRITER-64

But now we've added more power to Telewriter. Not just bells and whistles, but major features that give you total control over your writing. We call this new supercharged version Telewriter-64. For two reasons.

64K COMPATIBLE

Telewriter-64 runs fully in any Color Computer — 16K, 32K, or 64K, with or without Extended Basic, with disk or cassette or both. It automatically configures itself to take optimum advantage of all available memory. That means that when you upgrade your memory, the Telewriter-64 text buffer grows accordingly. In a 64K cassette based system, for example, you get about 40K of memory to store text. So you don't need disk or FLEX to put all your 64K to work immediately.

64 COLUMNS (AND 85!)

Besides the original 51 column screen, Telewriter-64 now gives you 2 additional high-density displays: 64 × 24 and 85 × 24!! Both high density modes provide all the standard Telewriter editing capabilities, and you can switch instantly to any of the 3 formats with a single control key command.

The 51 × 24 display is clear and crisp on the screen. The two high density modes are more crowded and less easily readable, but they are perfect for showing you the exact layout of your printed page, *all on the screen at one time*. Compare this with cumbersome "windows" that show you only fragments at a time and don't even allow editing.

RIGHT JUSTIFICATION & HYPHENATION

One outstanding advantage of the full-width screen display is that you can now set the screen width to match the width of your printed page, so that "what you see is what you get." This makes exact alignment of columns possible and it makes hyphenation simple.

Since short lines are the reason for the large spaces often found in standard right justified text, and since hyphenation is the most effective way to eliminate short lines, Telewriter-64 can now promise you some of the best looking right justification you can get on the Color Computer.

FEATURES & SPECIFICATIONS:

Printing and formatting: Drives any printer (LPVII/VIII, DMP-100/200, Epson, Okidata, Centronics, NEC, C. Itoh, Smith-Corona, Terminet, etc).

Embedded control codes give full dynamic access to intelligent printer features like: underlining, subscript, superscript, variable font and type size, dot-graphics, etc.

Dynamic (embedded) format controls for: top, bottom, and left margins; line length, lines per page, line spacing, new page, change page numbering, conditional new page, enable/disable justification.

Menu-driven control of these parameters, as well as: pause at page bottom, page numbering, baud rate (so you can run your printer at top speed), and Epson font. "Typewriter" feature sends typed lines directly to your printer, and Direct mode sends control codes right from the keyboard. Special Epson driver simplifies use with MX-80.

Supports single and multi-line headers and automatic centering. Print or save all or any section of the text buffer. Chain print any number of files from cassette or disk.

File and I/O Features: ASCII format files — create and edit BASIC, Assembly, Pascal, and C programs, Smart Terminal files (for uploading or downloading), even text files from other word processors. Compatible with spelling checkers (like Spell 'n Fix).

Cassette verify command for sure saves. Cassette auto-retry means you type a load command only once no matter where you are in the tape.

Read in, save, partial save, and append files with disk and/or cassette. For disk: print directory with free space to screen or printer, kill and rename files, set default drive. Easily customized to the number of drives in the system.

Editing features: Fast, full-screen editor with wordwrap, block copy, block move, block delete, line delete, global search and replace (or delete), wild card search, fast auto-repeat cursor, fast scrolling, cursor up, down, right, left, begin line, end line, top of text, bottom of text; page forward, page backward, align text, tabs, choice of buff or green background, complete error protection, line counter, word counter, space left, current file name, default drive in effect, set line length on screen.

Insert or delete text anywhere on the screen without changing "modes." This fast "free-form" editor provides maximum ease of use. Everything you do appears immediately on the screen in front of you. Commands require only a single key or a single key plus CLEAR.

...truly a state of the art word processor... outstanding in every respect.
— The RAINBOW, Jan. 1982

PROFESSIONAL WORD PROCESSING

You can no longer afford to be without the power and efficiency word processing brings to everything you write. The TRS-80 Color Computer is the lowest priced micro with the capability for serious word processing. And only Telewriter-64 fully unleashes that capability.

Telewriter-64 costs $49.95 on cassette, $59.95 on disk, and comes complete with over 70 pages of well-written documentation. (The step-by-step tutorial will have your writing with Telewriter-64 in a matter of minutes.)

To order, send check or money order to:

Cognitec
704 Nob Street
Del Mar, CA 92014

Or check your local software store. If you have questions, or would like to order by Visa or Mastercard, call us at (619) 755-1258 (weekdays, 8AM-4PM PST). Dealer inquiries invited.

RAINBOW
CERTIFICATION
SEAL

will continue to expand. I believe newer and better things are on the horizon. And, with the Color Computer on sale now, almost everyone has a chance to buy the best single machine available, dollar-for-dollar, on the market today. I hope you will do your friends and neighbors a favor and expose them to your CoCo in the days before and those following the holiday season.

I should also mention that RAINBOWfest in Irvine, Calif., will be in February. These shows are always a great deal of fun, as well as an opportunity to see the CoCo Community at its best. We have special rates at the Irvine Marriott for the show and hope that thousands of you will join us in sunny, Southern California for the midwinter meeting.

There are a host of seminars and lots of exhibitors planning to attend. So, check out the registration form in this month's issue and let us know that you will be with us.

Last, but certainly not the least, this is the most appropriate time of the year to say "thank you" to the tens of thousands of you who are members of the CoCo Community, who support THE RAINBOW, and who use and love your Color Computer. We started this magazine three and a half years ago with the idea of serving people who owned a CoCo, even though it didn't yet have that nickname.

We have been able to serve you, I feel, largely through *your* support of *us* — writing or calling us with suggestions; telling advertisers that you saw their ads in the magazine; submitting programs for publication; participating in our contests; and a whole host of other things. We truly could not have THE RAINBOW without each of you.

I hope this season of the year is a happy, healthy and prosperous one for you and yours. I look forward to being with you for an even better year ahead.

Happy Holidays and happy 1985.

— **Lonnie Falk**

One-Liner Contest Winner . . .

This shows how the notorious "*PMODE 4* color artifacts" create a color picture on your set from what is really a black and white video signal. It's also very nice to look at.

Mike Napolitan
W. Springfield, MA

The listing:

```
1 DIMA(11):W=20:FORC=4TO5:PMODE4
,1:PCLS:SCREEN1,1:GET(4,4)-(24,2
4),A,G:FORX=C TO228STEP2:Y=X*.75
:Z=174-Y:PUT(X,Y)-(X+W,Y+W),A,NO
T:PUT(X,Z)-(X+W,Z+W),A,NOT:PUT(X
,86)-(X+W,106),A,NOT:NEXT:FORT=1
TO500:NEXTT,C:FORB=1TO9999:NEXT
```

(For this winning one-liner contest entry, the author has been sent copies of both *The Rainbow Book Of Adventure* and its companion *Rainbow Adventure Tape*.)

Please hire the mentally retarded. They are sincere, hard working and appreciative. *Thanks! Phyllis.*

BUILDING JANUARY'S RAINBOW

Rainbow 'Excellent' Or 'Good' Say 98.8 Percent . . .
Some 94 Percent Have 32K, Most 64K . . .
Two-Thirds Frown On Three Column Listings . . .

When we decided to conduct a poll of those attending the Princeton RAINBOWfest, we knew that in order to get useful information from a survey you have to ask the right question — and that asking the right question is tricky business. Nonetheless, knowing full well we should consult an expert, such as the University of Cincinnati's Dr. Sam Sherrill, whose advice has appeared in RAINBOW, we instead gathered an ad hoc committee of those not busy on something else and developed questions based on a popular, if not professional, technique: "Hey, I know, let's ask how . . . "

Well, the results of the Princeton poll are in, and I believe we learned several things of value in addition to a great deal about taking future polls. In evaluating the results, we have kept in mind that those attending RAINBOWfest cannot be presumed to represent the entire RAINBOW readership. That is, as a group, those who came to the show were probably among the more dedicated users — likely more sophisticated in terms of hardware and software than readers who did not attend. After all, those attending traveled, on the average, some 136 miles to attend the Princeton show.

So, further caveats aside and with apologies to Sam, here (reported in percentages of those responding) are the highlights of what we found:

Present System:
Size:	4K	0.3
	16K	5.9
	32K	12.6
	64K	81.2
BASIC:	Color	4.7
	Extended	54.9
	Disk	40.4

Printer:
Make:	Radio Shack	54.2
	Other	45.3
Type:	Dot Matrix	86.6
	Letter Quality	8.1
	Color	2.8
	Other	2.5

Display:
Kind:	B&W TV	16.0
	Color TV	65.2
	TV/Video Monitor	8.8
	Computer Monitor	10.1

DOS Used:
Disk BASIC	84.0
OS-9	12.0
FLEX	2.5
Star-DOS	1.5

Software Most Likely To Buy:
Utilities	23.0
Games	19.3
Word Processing	17.5
Home Uses	14.1
Education	13.0
Art/Graphics	11.0
Business	1.5
Music	0.6

Items Most Interested In Buying:
Disk Drives	17.8

More Memory	13.4
Printer	12.1
Modem	10.2
Speech/Sound Synthesizer	10.0
Monitor	7.5
Graphics Tablet	6.2
New Operating System	5.6
Expansion Unit	5.1
CP/M Adapter	4.1
Joysticks	3.4
Remote Device Control	2.8
Videodisc Player	1.9

I Rate THE RAINBOW:
Excellent	80.8
Good	18.0
Fair	1.1
Poor	0.1

Average Number of Recent Purchases Through RAINBOW Ads:
Total	6.8

I would like to see THE RAINBOW offer more:
Utilities	18.6
Machine Language Info.	15.7
Hardware Projects	12.6
Educational Programs	11.0
Business Programs	10.5
Reviews	10.2
Tutorials	10.1
Games	8.6
Other	2.9
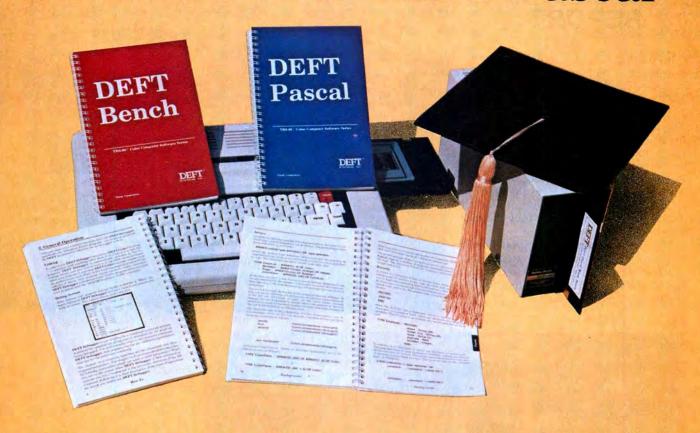
I think your Product Reviews are:
Just right	72.5
Too short	4.9

RAINBOWfest REPORT

Bright lights and a crush of people crowded the Computize booth throughout the run of RAINBOWfest-Princeton, Sept. 28-30, as Ken Klosinski and his crew made "video snapshots" of show-goers using the Graphicom Video Digitizer. Meanwhile, Frank Hogg and his helpers were showing off "Nomad," a little personal robot that rode around under CoCo control. Bob Rosen, of Spectrum Projects, seemed to have more show special signs than anyone could be expected to read. But, it was our RAINBOW readers who really took the cake — birthday cake.

We've been sworn not to repeat Lonnie Falk's age, so, we'll only state it one time that he'd just turned 42. He wanted to celebrate the occasion with those attending RAINBOWfest — thus, cake for all.

1

2

3

4

5

6

We have no statistics on how many dozen eggs, pounds of sugar or cups of flour went into the cake that served thousands, but if it's numbers you want, we have compiled the results from the survey we made of those attending "CoCo's very own show." For the highlights, see Jim Reed's column on Page 16 of this issue. A name was drawn from those responding to the survey and Michael J. Clancy, a real estate consultant from Jessup, Md., won himself a color graphics printer, just for filling out the survey form.

Was this a show for computer die-hards? Well, while Julie A. McGee, director of software development for Tandy Home Education Systems, drew a full house as keynoter at the CoCo Community Breakfast, Saturday morning, one of the most celebrated students from nearby Princeton University, Brooke Shields, went almost unnoticed by the CoCo crowd as she brunched Sunday morning in the hotel coffee shop.

As with all our RAINBOWfests, the Princeton show ran at an accelerated pace in order for everyone to meet as many people, exchange as much information, and interact as much as possible in our brief time together. And, nobody worked harder than Jerry Behler and the members of the Penn-Jersey Color Computer Club, who helped us publicize and run the show. Our thanks, Penn-Jersey, for helping, and our thanks to all the many CoCo clubs and individuals whose attendance makes these shows possible.

Next stop, Irvine, Calif. We do hope you'll join us.

7

8

9

10

11

12

13

The Pictures: (1) Al Alberta draws a crowd at the PBJ, Inc. booth. (2) Jack Torres of Computer Plus shows a new printer to Hot CoCo publisher Jeff Detray (blue shirt). (3) A computing family takes note of a bargain at one of the many vendor display booths. (4) Claire Manfredonia apparently caught her fingers in a joyport during a voltage surge. (5) Programmer Bill Dunlevy *(Cashman, Time Bandit)* worked with the MichTron crew. (6) One of Bill's programs gets a workout from a young challenger, one of 8,600 attending the Princeton show.

(7) Tandy's Julie McGee with Rainbow's Lonnie Falk at the CoCo Community Breakfast. (8) Also at the head table, Cheryl Blyn, of Computer Island, (left) with Sugar Software's Susan Davis. (9) Nearly 500 attended the breakfast, Saturday morning. (10) While the CoCo crowd was upstairs in the exhibit hall, Brooke Shields and her mother went *almost* unnoticed at brunchtime Sunday in the Hyatt coffeeshop. (11) Artist Paul Hoffman helps out at Owl-Ware. (12) Even the merchandise was friendly at the Penn-Jersey display. (13) Al Timme of Hard Drive Specialists (formerly Compukit Corp.).
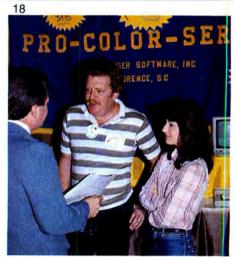
(14) RAINBOWfest is a good place to make a close examination of hardware products. (15) The milling crowd of CoCo enthusiasts filled the Regency Ballroom at the Hyatt. (16) Frank Hogg discusses *Dynacalc* with visitors to the FHL booth. (17) Bob Beckett responds to a roving CoCo crew. (18) It's hard to beat this CoCo artistry. (19) Paul Kush fields a question at Derringer Software. (20) Pat Endicott had a bargain corner at Endicott Software.

14

17

18

15

19

20

16

THE BIG 3

DYNACALC ®

Telewriter-64 ™

Pro-Color-Series ™

THE LOGICAL CHOICES

It Is Better To PUT Than To GET

By Alexander B. Trevor

A little experience with Extended Color BASIC graphics will make it obvious that by far the fastest way to place graphics elements on the screen is with the *PUT* statement. In fact, it is almost the only way to implement any kind of animation on the Color Computer in BASIC. Most descriptions of *PUT* tie its use to the *GET* statement, although this does not have to be the case, as we shall see. (Radio Shack's *Going Ahead With Extended Color BASIC,* pages 67-71, and *Color Computer Graphics,"* by William Barden, Jr., pages 143-154 are two examples). This is not too surprising, since the two statements are intended to be complementary. The idea is the *GET* will copy graphics data out of a specified rectangular area on the graphics screen, and save it in an array. The *PUT* statement allows you to place the data back on the screen at the same or another location.

In a typical game application, a number of "sprites" (such as rockets, robots, lunar landers, etc.) will be defined at the beginning of the program and then used with *PUT* statements throughout the game. The usual method of defining the sprites is to use the *DRAW* command to (slowly) draw each sprite on the graphics screen; then, each sprite is stored in a separate array with the *GET* command.

There are several problems with this method, though none of them may be serious in many applications: first, the *DRAW* command must be done on a separate graphics page, or it will deface anything that is already on the page. This is an important consideration for graphics editors, which may allow you to work on an image already in graphics memory, but not important in any program that clears the graphics page upon start up. Second, if there are many items to be drawn, the setup process can cause a noticeably long delay. Third, after *GETting* graphics into an array the contents of the array become "invisible," and

cannot be printed to the screen, tape or disk. The *GET* command purposely trashes part of the array header to cause this undesirable side effect.

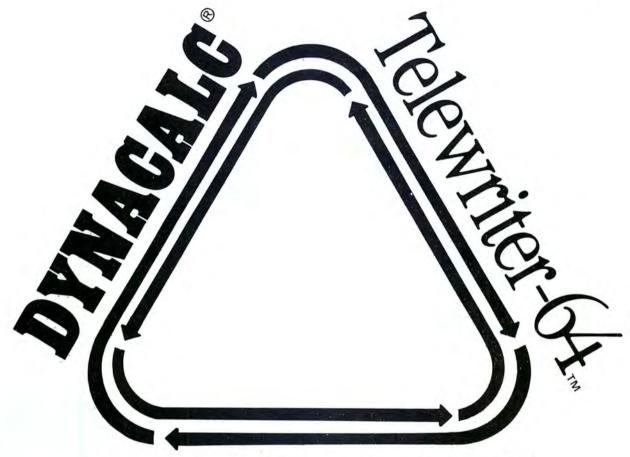The method described here avoids these problems through an alternate way to load arrays with graphics data that can be used in a *PUT* statement. To use this technique, it is necessary to understand the format that data is stored on the screen, and also the nature of arrays. Rather than attempt to describe all the possibilities, I will limit this discussion to *PMODE 4*, the high resolution mode consisting of 256 horizontal elements (pixels) and 192 pixels vertically. The technique is easily extended to other graphics modes.

In *PMODE 4*, each byte in the graphics page contains eight pixels along the horizontal. Since there are 256 pixels on each line, it takes 32 bytes to hold the first line. There are 192 lines in the entire picture, or 6,144 bytes. Data for *GET* and *PUT* is stored in a similar scheme. If you *GET* a 5 x 4 area into an array, you are storing 5 x 4, or 20 pixels. In this case one pixel requires one bit of storage. The pixels are stored tightly packed in the array, with the upper left hand pixel stored as the most significant bit of the zero byte of the array. Since each byte can contain eight bits, this graphic will require only three bytes. For example a lowercase 'v' might appear on the screen like this:

Graphics Screen Area (5 horiz x 4 vert)

```
            Column
        1     2     3     4     5
Row 1:  1     .     .     .     1
Row 2:  1     .     .     .     1
Row 3:  .     1     .     1     .
Row 4:  .     .     1     .     .
```

As stored in the array;

```
Byte 1:  1 0 0 0 1     1 0 0
         (row 1)    (row 2, 1-3)
```

(Alexander Trevor, who holds a master's degree in electrical engineering, is executive vice president of Computer Resources at CompuServe, and is a member of the IEEE and the ACM. In his "spare time" he writes software for CompuServe's DEC-10 mainframes and for a variety of microcomputers.)

Byte 2: 0 1   0 1 0 1 0   0
(row 2, 4-5) (row 3) (row 4, 1)

Byte 3:   0 1 0 0   0 0 0 0
(row 4, 2-5) (fill)

Arrays in Extended Color BASIC consist of elements that are five bytes each. Five bytes are necessary to hold a floating point number with the precision used in ECB, but the five bytes have no bearing on the graphics use of arrays except to make dimensioning and loading more confusing. In the above example, the three bytes will fit easily into the five bytes that will be allocated to a single array element.

How do we get three bytes of data into the first three bytes of a five byte integer array element? First, we must find out the address of the array element with the *VARPTR* statement — one of the less frequently used BASIC commands. Then, we *POKE* the data into the memory locations reserved for the array. That's all there is to it! The array can now be used in a *PUT*.

To see how this works in practice, let's follow through the example of a lunar lander sprite as shown below. The sprite is drawn in a rectangular area on the graphics screen eight columns wide and seven rows high:

| | Column | | | | | | | | Hex |
	0	1	2	3	4	5	6	7	Value
Row 1:	.	.	1	1	1	1	.	.	2C
Row 2:	.	1	1	1	1	1	1	.	7E
Row 3:	.	1	1	1	1	1	1	.	7E
Row 4:	.	1	1	1	1	1	1	.	7E
Row 5:	.	.	1	.	.	1	.	.	24
Row 6:	.	.	1	.	.	1	.	.	24
Row 7:	.	1	1	.	.	1	1	.	66

In this example, it is particularly easy to determine the value to be poked into the array because the graphics area is exactly eight pixels wide. Thus, an entire row of pixels fits exactly into one byte of the array. In cases where the rectangle width is not a multiple of eight (such as the example of the lowercased 'v' above), you may wish to draw the graphic on the screen using any method (*DRAW* command, or graphics editor), then use the program given in Listing 1 to print out the appropriate values.

Listing 2 is a complete example for the *"PUT without GET"* technique. In Line 20, four graphics pages are reserved. Line 30 sets the horizontal and vertical size of the array. These values (HSIZE and VSIZE) are then used to *DIM*ension the array LL. HSIZE*VSIZE is the number of pixels; this is divided by eight bits per byte and five bytes per array element (i.e., 40), and rounded to the next integral byte. A subroutine to load the graphics array LL is called from Line 50, while lines 60-90 simply *PUT* the element on the graphics screen. The symbol setup routine (lines 100-170) is the key to the technique. In Line 110, all variables to be used within the subroutine are referenced. *This is absolutely necessary* in order to prevent the location

of the array from changing after the *VARPTR* statement. If an undefined variable is encountered by BASIC, all variables are relocated in memory, invalidating the address returned by the *VARPTR* statement in Line 120. With an accurate address for the zero element of LL in variable P, the data is *POKE*d into each byte of the array in Line 150. In a variation of this method, the graphics data can be read in from disk or tape rather than from the *DATA* statements.

I have found this technique particularly useful for programs that use a number of sprites, and in which it was desirable to reduce the setup time to a minimum. Next time you want to speed up a graphics program, *PUT* something you didn't *GET*. You'll be rewarded with a faster program.

Listing 1:

```
10 ' PRINT DATA FOR A "PUT"
20 ' USE AFTER DRAWING GRAPHIC
30 X=100: Y=100 'UPPER L CORNER
40 HSIZE=8 'SET HORIZONTAL SIZE
45 VSIZE=7 'SET VERTICAL SIZE
50 DIM LL(HSIZE*VSIZE/40+1)
55 'VARIABLES MUST BE REFERENCED
56 'BEFORE CALLING VARPTR
60 I=0: P=0: J=0
70 GET(X,Y)-(X-1+HSIZE,Y-1+VSIZE
),LL,G
80 P=VARPTR(LL)+12
90 FOR I=0 TO HSIZE*VSIZE/8-1
100 IF J=0 THEN PRINT:PRINT"DATA
"; 
110 PRINT HEX$(PEEK(P+I));
120 IF J>6 THEN J=0 ELSE J=J+1:P
RINT",";
130 NEXT
140 PRINT"0"
```

Listing 2:

```
10 ' GRAPHICS PUT WITHOUT GET
20 PCLEAR 4: PMODE 4
30 HSIZE=8:   VSIZE=7
40 DIM LL(HSIZE*VSIZE/40+1)
50 GOSUB 100
60 PCLS:       SCREEN 1,1
70 X=100:      Y=100
80 PUT(X,Y)-(X-1+HSIZE,Y-1+VSIZE
),LL,PSET
90 GOTO90
100 'SYMBOL SETUP SUBROUTINE
110 I=0: T$="    ": P=0   'MUST
USE VARIABLES BEFORE VARPTR!
120 P=VARPTR(LL(0))
130 FOR I=0 TO HSIZE*VSIZE/8-1
140 READ T$
150 POKE P+I,VAL("&H"+T$)
160 NEXT
170 RETURN
180 'LUNAR LANDER SYMBOL
190 DATA 3C,7E,7E,7E,24,24,66
```

16K
ECB

Home Sweet Home

By Marlene Fearing
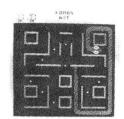
This program draws a house, a sun and a garage. It opens and closes the garage door, the front door opens and a figure appears and waves. Afterward, the door closes, the grass grows, and smoke comes from the chimney.
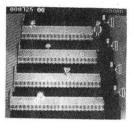
This is the first computer program I wrote after getting my computer. I hope it will encourage others to experiment with graphics and animation; it was a lot of fun to create. This program will work with 16K Extended BASIC with tape, or with a disk drive system. Just type it in and watch it draw.

(Marlene Fearing is a student at Pima Community College in Tucson, Ariz., where she is studying for her A.A.S. as a small business computer specialist.)

The listing:

```
1  '
2  '******************************
3  '*                            *
4  '*      EXECUTIVE HOUSE        *
5  '*      MARLENE FEARING        *
6  '*      812 S. PLUMER          *
7  '*      TUCSON, ARIZ. 85719    *
8  '*                            *
9  '******************************
```

```
10 PMODE 3,1
11 PCLS (3)
12 SCREEN 1,1
13 '    DRAW MAIN HOUSE
14 LINE (32,180)-(152,88),PSET,B
15 LINE (28,68)-(156,88),PSET,BF
16 PAINT (32,72),2,4
17 LINE (152,91)-(240,180),PSET,
B
18 LINE (170,108)-(226,180),PSET
,B
19 '        DRAW THE SUN
20 CIRCLE (204,22),10,2
21 LINE (44,104)-(68,140),PSET,B
22 LINE (124,104)-(148,140),PSET
,B
```

```
23 LINE (82,104)-(112,180),PSET,
B
24 LINE (76,52)-(100,68),PSET,BF
25 CIRCLE (84,140),2,2
26 PAINT (44,160),1,4
27 PAINT (169,176),1,4
28 CIRCLE (200,176),3,2
29 PAINT (204,22),1,2
30 PAINT(56,120),2,4:PAINT (133,
120),2,4
31 LINE (56,104)-(56,140),PSET
32 LINE (136,104)-(136,140),PSET
33 '          TO OPEN AND CLOSE GAR
AGE DOOR
34 FOR X=1 TO 500:NEXT X
35 PAINT (190,179),2,4
36 FOR X=1 TO 1500:NEXT X
37 PAINT (176,110),4,4
38 CIRCLE (200,176),3,2
39 LINE (0,180)-(255,191),PSET,B
40 '         TO OPEN AND CLOSE FRONT
 DOOR AND FIGURE TO WAVE AND GO
BACK INSIDE
41 LINE (92,112)-(92,190),PSET
42 LINE (92,190)-(112,180),PSET
43 PAINT (185,190),1,4
44 PAINT (10,185),2,4
45 LINE (92,112)-(112,104),PRESE
T
46 PAINT (185,190),2,4
47 LINE (92,112)-(92,190),PRESET
48 LINE (92,190)-(112,180),PRESE
T
49 LINE (0,255)-(255,180),PSET,B
50 FOR X=1 TO 120:NEXT X
51 LINE (102,112)-(112,104),PSET
52 LINE (102,112)-(102,190),PSET
53 LINE (102,190)-(112,180),PSET
54 PAINT (96,124),1,4
55 CIRCLE (96,124),7,0
56 LINE (96,130)-(96,164),PSET
57 LINE (96,164)-(84,179),PSET
58 LINE (96,140)-(84,140),PSET
59 LINE (96,164)-(102,179),PSET
60 LINE (96,140)-(102,140),PSET
61 LINE (86,140)-(86,130),PSET
62 FOR X=1 TO 300:NEXT X
63 LINE (86,140)-(86,120),PRESET
64 FOR X=1 TO 500:NEXT X
65 LINE (86,140)-(86,130),PSET
66 FOR X=1 TO 150:NEXT X
67 LINE (86,140)-(86,130),PRESET
68 FOR X=1 TO 150:NEXT X
69 LINE (86,140)-(86,130),PSET
70 CIRCLE (96,124),7,1
71 LINE (96,130)-(96,164),PRESET
72 LINE (96,164)-(84,179),PRESET
73 LINE (96,140)-(84,140),PRESET
74 LINE (0,180)-(255,180),PSET
75 LINE (96,164)-(102,179),PRESE
T
76 LINE (96,140)-(102,140),PRESE
T
77 LINE (86,140)-(86,130),PRESET
78 LINE (102,112)-(112,104),PRES
ET
79 LINE (102,112)-(102,190),PRES
ET
80 LINE (102,190)-(112,180),PRES
ET
81 LINE (0,180)-(255,191),PSET,B
82 CIRCLE (86,140),3,3
83 COLOR 2,1
84 PAINT (30,188),2,4
85 PAINT (232,188),2,4
86 '
87 '       TO DRAW GRASS
88 '
89 POKE 65495,0
90 DRAW "BM0,180;R1;U8;R2;D8;R2;
U10;R2;D10;R3;U12;R2;D12;R3;U5;R
2;D5;R3;U5;R2;D5;R2;U3;R2;D2;R2;
U4;R2;D4;R2;U3;R2;D3"
91 DRAW "BM238,180;U10;R2;D10;R3
;U8;R2;D8;R4;U6;R2;D6;R2;U8;R2;D
8;R1"
92 '
93 'SMOKE STARTS HERE
94 '
95 X=82:Y=52: 'CIRCLE CENTERPOIN
T
96 SP=0:EP=0  'CIRCLE RADIUS
97 FOR R=1 TO 35 STEP .05  'CIRC
LE RADIUS
98 EP=EP+.02: IF EP>1 THEN EP=0
99 CIRCLE (X+R,Y-R),R,1,1,SP,EP
100 NEXT R
101 '
102 '          TO TURN BACKGROUND TO
 NIGHT
103 '
104 PMODE 4,1
105 SCREEN 1,0
106 CIRCLE (204,22),10,5
107 PAINT (208,22),5.5
108 '          REDRAWN SMOKE STARTS
 HERE
109 X=82:Y=52:  'CIRCLE CENTERPO
INT
110 SP=0:EP=0:  'CIRCLE RADUIS
111 FOR R=1 TO 35 STEP.05 'CIRCL
E RADUIS
112 EP=EP+.02:IF EP>1 THEN EP=0
113 CIRCLE (X+R,Y-R),R,1,1,SP,EP
114 NEXT R
115 POKE 65494,0
116 GOTO 10
117 END
```

16K
ECB

EZ-Graphics — '85 Style

By Fred B. Scerbo
Rainbow Contributing Editor

Editor's Note: If you have an idea for the Wishing Well, submit it to Fred c/o THE RAINBOW. Remember, keep your ideas specific, but don't forget that this is BASIC. All programs resulting from your wishes are for your use but remain the property of the author.

Each of us often spends a great deal of time making New Year's resolutions which are quite often forgotten within several days of the pledge. Although I have some resolutions which I will try to work into my daily routine, one resolution which I really wanted to make was a pledge to those of you who still have only 16K Extended Color BASIC and have been left along the roadside during our last few "Wishing Well" articles. (The last three have all been for 32K.) While this does not mean that you won't be seeing the most requested sequels, such as "Rockfest II" and "Baseball Fever II," I figured that this would be a good time to offer all of you some shorter listings that will equally satisfy everyone from 16K to 64K. This will be a great relief to all of you who will want to key these listings

(Fred Scerbo is a special needs instructor for the North Adams Public Schools. He holds a master's in education and published some of the first software available for the Color Computer through his software firm, Illustrated Memory Banks.)

in but may be too pressed to hammer in the really long ones during the hectic rush of the holiday season. Also, some of you may have just gotten a CoCo for the first time during the holiday season and odds are that more of you got 16K than 64K.

So, as a little New Year's gift for all of you CoCo enthusiasts, here are two short listings which I dare any friends you may have to equal in as few lines on an Apple II or Commodore 64. These are strictly for the fun of creating sharp graphics. Next month we will get back to some more serious wishes.

Who You Gonna Call?
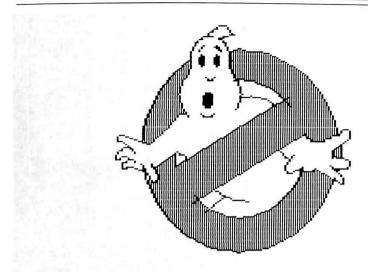
Probably one of the most successful motion pictures of the past summer movie season was the comedy, *Ghostbusters*. Besides being a funny movie, this cinematic effort has as one of its offsprings a symbol which is quickly becoming as common as "Rubik's Cube," Michael Jackson and "Cabbage Patch" dolls. Recently, I have seen this particularly catchy no-ghosts logo showing up on everything from T-shirts to bumper stickers. Well, here's one more place you can look to see this omnipresent poltergeist: on the screen of your CoCo.

Why even do this? Well, as I have said before, young CoCo programmers often take great pride in being able, with just a few program lines, to create graphics which are easily recognized. This eventually will serve to stimulate

even the most inexperienced programmer to learn more, and eventually create a program which others can benefit from as well.

Therefore, Listing 1 is an extremely short program which will recreate the *Ghostbusters* logo in rather dramatic detail before your eyes. The actual body of the listing which draws the ghost and the slashed circle is only about 13 lines long. As I just mentioned, I doubt any Apple or Commodore programmers will be able to match this graphic on their machine in as few lines. This just gives you one more weapon to use in convincing others that you made the best choice when you chose a Color Computer.

The actual graphic uses *PMODE 4* with an overlay of *PMODE 3* (without using the *SCREEN* command). The ghost and slash are formed by a combination of semi-circle and *DRAW* commands, accounting for how efficiently this BASIC code can be written. You would need a little trigonometry to get similar results on the other computers.

One difference you will notice this time around is that I have used the Reset button to control the occurrence of red. When you *RUN* the program, if the screen is not red, press Reset and *reRUN* the program until it is red. Once it is red, press the ENTER key to draw the graphics. Most of my other programs usually offset a pixel to control the colors without using Reset, but since this was such a short listing, I figured

that the Reset was the quickest route to follow.

A final word should be mentioned here before we move on to our second listing. The actual *Ghostbusters* symbol is the property of Paramount Pictures which holds all rights for its commercial use. Therefore, this listing is for your own personal home use for the fun of it, and may not be used for any promotional purposes. (For example, if any of you were thinking of writing your own *Ghostbusters* game, you could not use this graphics or the logo as part of your effort.) However, no harm should come from using this listing for the fun of learning more about how your CoCo's graphics commands work. Consider it an educational experience.

Therefore, enjoy this little graphics gem, and let me know if you have any ideas for other similar efforts I might be able to share in the "Well."

Sharing The Wealth (Of Graphics Skills)

In the last two installments of "The Wishing Well," I shared with you a technique of using checkerboard pixel patterns to create extra colors in *PMODE 4* (and *PMODE 3* as well). We saved these patterns in GET-PUT arrays, and painted them on the screen using the *OR* command found with *PUT*. Using this technique, any area which has previously been painted black will be filled in with the color found in our array. For a more detailed explanation of how these colors are generated, refer to last month's article.

There was one small drawback with the method used in last month's issue. While the technique was completely effective for what we wanted to accomplish with those football helmet graphics, the routines were not designed for you to easily use if you wanted to use the extra colors in your own drawings. As I promised, I have come up with a way that you could use them easily without

having to do a great deal of graphics gymnastics. The method I have listed here will be a piece of cake to anyone who knows how to use the *LINE* command found in your Color Extended language.

Another limitation found in last month's version was the fact that the array covered nearly the full width of the screen. This would mean that your graphics would have to be drawn and colored in a fashion that would not allow you to have a yellow object next to a purple object, since the arrays would overlap each other on the same level. Therefore, these new routines have set up arrays which are only 16 pixels wide and two pixels deep. This allows greater flexibility in this type of painting when more than one color is desired on the same left to right level. It also takes up less than 10 percent of the memory required to do it the original way. The routines used for "Football 1 and 2" are much faster than this technique. Since speed was more important than flexibility, those were written with speed in mind. As always, you have the classic trade-off. Speed and flexibility are inversely proportional. This month we will emphasize flexibility instead, while sacrificing speed.

Therefore, the BASIC code required to efficiently and easily use these colors has been written to be part of the first 25 lines of your program. If you wish to draw graphics using them, you would simply start your own program lines at Line 100. There is a special syntax which I have developed to handle the colors which I will explain in just a few lines. Simply put, it is a new way of coloring, but at the same time will be very familiar to you.

Actually, the most difficult part of writing this program was thinking of what to draw as a graphics to demonstrate the routines. I didn't want to do a rock logo because those will be showing up in a couple of months, and there was

no reason to let the wind out of my sails for that one yet. Secondly, the colors would be of no use for the *Ghostbusters* graphics listed here. Besides, that would make it longer, defeating the whole purpose of doing the logo in the first place.

When the idea for what to draw finally came to me, I wanted to kick myself for not having thought of it in the first place. What is one of the first paintings or drawings that an aspiring young artist starts with? Why, of course, the answer was a bowl of fruit! Sure, it may not be as dramatic as the car from the ZZ Top logo of a few months ago, but it would give me the possibility of drawing and using more than one of these colors side by side.

With this in mind, I developed a set of seven additional color patterns to be included in the arrays. These patterns are set in lines 11-14 and put into the arrays in Line 15. The colors and their corresponding Syntax letters are listed below. Remember, the actual color may depend on how accurately your TV set tint is adjusted to red and blue.

Y = Yellow
B = Light blue
G = Gold or orange
S = Silver or gray
P = Purple (dark)
L = Lime or dark blue
V = Violet

The lime color is not really so much of a green, but depending on your tint, it is about as close as we can get. You will notice that I mentioned that the letters are part of the syntax. You will actually use these letters to call the colors as you need them.

Remember how I mentioned that the syntax would be familiar to anyone who knew how to use the *LINE* command? As you may recall, the syntax for *LINE* is:

LINE (x1,y1)—(x2,y2),PSET

The variables x1,x2,y1 and y2 are used to define the starting and end points of a line (or box) using x and y coordinates on a field of 256x192 pixels. Therefore, if you wanted to draw a box with opposite coordinates of 10,10 by 20,20 you would write:

LINE(10,10)—(20,20),PRESET,BF

which would give you a box 10 pixels square painted in black (because of BF which means box filled). To paint this box with our new colors, you would use the same coordinates. These coordinates are placed in a STRING 1 call PAINT STRING which is identified as PT$ in these routines. The syntax for

What goes well with *the Rainbow?*

Rainbow On Tape!

We call it the other side of THE RAINBOW and we may have to raise the price just to call your attention to it. With as many as two dozen programs every month, *Rainbow On Tape* is a luxury service at a bargain basement price.

What is it? *Rainbow On Tape* is a monthly, cassette tape adjunct to THE RAINBOW and it's brimming with all the programs (those over 20 lines long) that fill the pages of the magazine. All you do is pop the cassette in your tape recorder and they're ready to run. No more lost weekends —or weeknights — typing, typing, typing. With *Rainbow On Tape*, you can read the article in the magazine then, in seconds, you load it up and run it.

Yes, *Rainbow On Tape* is brimming with the programs that fill THE RAINBOW's pages each month. And, yes, you could type them in yourself, as many people do. But all of them? Every month? There simply isn't enough time.

Isn't it time your CoCo became a fulltime computer instead of a typewriter. Think how your software library will grow. With your first year's subscription, you'll get almost 300 new programs: games, utilities, business programs, home applications — the full spectrum of THE RAINBOW's offerings without the specter of keying in page after page and then debugging.

Rainbow On Tape — the "meat" of THE RAINBOW at a price that's "small potatoes." Food for thought. To get your first heaping helping, just fill out and return the attached reply card. No postage necessary.

Discover the other side of THE RAINBOW. It's not only a time-saver, it's the key to a whole new outlook!
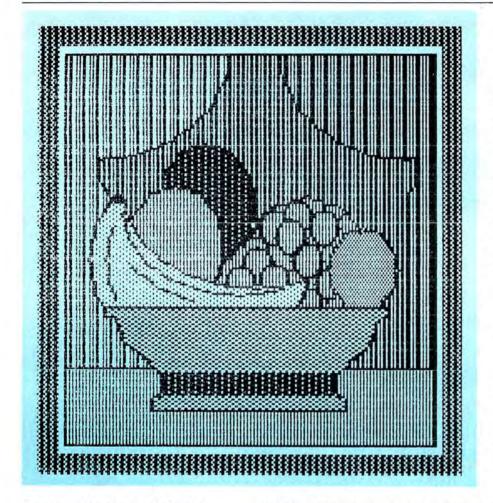
these coordinates would thus be:

PT$="Y010,010—020,020"

The Y stands for the color yellow. The next three digits are the coordinate for our x1 coordinate followed by a comma, with the next three digits being the y1 coordinate. We then use a dash and use three digits for x2, a comma, and three digits for y2. The coordinates for a box such as:

LINE(10,20)—(30,40),PSET,BF

would be:

PT$="Y010,020—030,040"

In each case, we use this PAINT STRING by following it with the following command:

GOSUB 17

which takes care of the painting. You will notice that even though the numbers we are using are only two digit numbers, we must use three digits such as 010 for 10, or 006 for the number 6. This is necessary because the PAINT STRING is analyzed in lines 17 and 18 to determine the coordinates and

colors. This was much easier than to have you type in the values for five separate variables. If you accidentally use two digits rather than three, the painting will not take place. There will not be an error message. There will just be no painting. Thus, our syntax for PAINT STRING is:

PT$="Color,Left Corner,Top Corner—Right Corner,Bottom Corner"

followed by GOSUB 17. All corner coordinates must be three digits. As you can see, if you know how to use LINE, you will have no trouble using PAINT STRING.

If you RUN the second listing called Seven More PMODE4 Colors, you will have a very nice, framed painting of a bowl of fruit with drapes in the background. You will be pleased to see that this does make a very nice graphics to use for showing the colors available on your CoCo.

I did not use all seven colors here. Rather, I used just a few so you would get the idea. To get a nice curtain or draped effect, I used POKE 178,x to give a little added realism. Remember, the technique used for this is to use a value between zero and 255 with the

POKE and the PAINT using:

PAINT(x,y),,1

to get your striped colors. This was described many months ago in THE RAINBOW.

Lines 310 to 380 are designed to let you change PMODEs and SCREENs to see how these patterns look under different combinations. Hitting the ENTER key will flip through the various combinations. I have also used Reset to control red in this program. If the screen is not red when you RUN it, press Reset until it is and then hit ENTER to continue. Again, I felt that this would be preferable to my other method since the Reset route is very popular, and you might find it easier since you may want to use these routines yourself.

Let's say you have RUN this program and now want to use these colors for your own graphics. Load in the program and type:

DEL 110—

and hit ENTER. This will delete all following lines keeping the routines intact. It will also set your screen for PMODE 4 with PMODE3 colors. You may wish to alter Line 100 to suit you needs. Since I have already used a number of variables in the routines, you will want to take care not to use these same variables! Here is a list of the variables which you should avoid:

R,B,X,Y,G,S,P,L,V,LC,RC,TC, BC,YY,ZZ

and the string variables:
XX$ and PT$
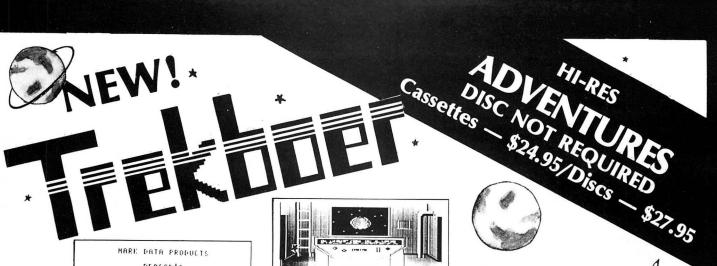
The variables R and B stand for red and blue, and you may substitute them in the program to suit your needs. I have chosen instead to use the values of 3 and 2 in the program so as to not bury you in variables.

If you wish to use these routines, you may renumber them, but you must leave the REM statements intact since this program is under copyright. Feel free to create using these techniques, but remember to give credit where credit is due! That's what makes it possible to share these techniques with you.

Let's try a little experimenting so you can see how this really works. Delete the first lines as I mentioned and type in the following new lines:

```
110 CIRCLE(128,96),60,1,.9
120 PAINT(128,46),1,1
130 PT$="Y068,042—190,150":
GOSUB17
140 LINE(68,42)—(190,150)
,PRESET,B
```

1000 GOTO1000

This will draw a circle, paint it black, PAINT STRING it yellow, and surround it with a box that shows the area actually covered by the array. Thus, if you have an irregular shaped object and paint it black, you can fill it in with these colors just as you would with PAINT because we are using *OR* which checks to see if a pixel is set, and if it is not, it sets it to the pattern. You may also need to redraw around the object since the color will fill in any area it overlaps, which is black.
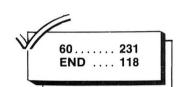
Although these new smaller arrays do mean you can have adjacent colors, be careful not to make the items too close together, as I tried to be sure of with the fruit. With a little experimentation, you will be painting with ease in no time at all. (In fact, you will most likely be seeing the routines and variations on them in upcoming graphics wishes. I mean, why shouldn't I take advantage of this easier method as well?)

In playing with the sample I have just given you, change the letter for the color in Line 130. This will give you a better idea of how to control the colors. Probably the hardest thing to color this way would be concentric circles. If you can handle that one, you can handle most any graphics. Yes, it can be done, but I won't show you how now. Let's see you try it yourself.

Conclusion

When I think of the types of graphics people originally got excited about when Color Extended came to the CoCo, and I see how far we have come with the very same machine, I can only imagine that things will continue to get more exciting. I started with Color BASIC with a $499 16K machine and thought that block graphics were great back in 1981. Who would ever think we would be milking such detail out of this machine without really changing the original language? I'll keep searching for ways to make these things better. You just keep feeding me ideas.

Happy New Year and Happy Programming!

Listing 1:
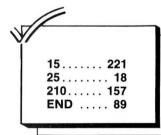
```
1  '******************************
2  '*        GHOSTBUSTERS LOGO      *
3  '*          BY FRED B. SCERBO    *
4  '* 149 BARBOUR ST.N.ADAMS.MA*
5  '*      COPYRIGHT (C)  1984      *
6  '* LOGO IS THE PROPERTY OF       *
7  '* OF PARAMOUNT PICTURES &       *
8  '* IS FOR YOUR HOME USE ONLY*
9  '******************************
10 PMODE4,1:PCLS1:SCREEN1,1:PMOD
E3:R=3:PCLS3
20 IFINKEY$<>CHR$(13)THEN20
30 PCLS4
40 CIRCLE(100,40),20,1,.9,.46,.0
5:DRAW"C4BM100,24NU2R6DR2C1R4ER2
M+6,-1R2L3H6L6G2L2G2LG2DR3"
50 CIRCLE(100,62),30,1,.66,.41,.
63:CIRCLE(100,62),30,1,.66,.9,.1
:DRAW"BM-22,+24M+2,-4BR36M+2,+4F
2BM-16,-20F4M-6,-3R2BL12BUG4BD4B
RD4F2U8F2D5BR9BUNU4F2U8F2D5BD8BL
4G2L4H2BD12BL2D6F2U10R2D10R2U10F
2D6BD6BR2G2L8H2BL16BU2F4H2L6"
60 DRAW"M-12,+10M-16,+10M-24,-6L
4G2D2R4M+10,+6NF2H2L6G2L2G4D2R4E
2R4F2R4F2H2L4G2L4D4R4ER4M+10,+2R
4NH4M+6,+8R4E2U2M-6,-8D2R4M+20,-
10F2R4F2D"
70 CIRCLE(138,80),30,1,.6,.69,.9
:CIRCLE(136,112),42,1,.5,.25,.4
80 DRAW"BM114,130M-18,+10"
90 DRAW"BM166,100F4R2F2NE4G4D2F2
R4M+9,-3F2R9E2M+24,+10R2U4M-12,-
8M+3,+2R4NDR2NDR8E2U2H2M-14,-2ND
U4M+8,-4ND8M+8,-4U4H2L4G4L2NU4L4
```

G2L4G2L12H2L2H2"
```
100 CIRCLE(128,98),92,1,.85,.52,
.675:CIRCLE(128,98),92,1,.85,.74
,.98:CIRCLE(128,98),92,1,.85,.05
,.48
110 CIRCLE(128,98),62,1,.85,.74,
.9:CIRCLE(128,98),62,1,.85,.05,.
39
120 DRAW"BM68,110NU8M+104,-46BF2
0BD4BL4M-104,+46R2"
130 PAINT(78,36),R,1:PAINT(134,2
4),R,1:PAINT(218,100),R,1
140 PAINT(2,2),1,1:PAINT(134,50)
,1,1:PAINT(72,106),1,1:PAINT(92,
136),1,1
150 GOTO150
```

Listing 2:

```
1  '******************************
2  '* SEVEN MORE PMODE4 COLORS *
3  '*        BY FRED B. SCERBO     *
4  '* 149 BARBOUR ST.N.ADAMS.MA*
5  '*      COPYRIGHT (C)  1984      *
6  '******************************
7  CLEAR1000:R=3:B=2
8  PMODE4,1:PCLS1:SCREEN1,1:PMODE
3:PCLS3
9  IFINKEY$=CHR$(13)THEN11ELSE9
10 'START COLOR SET
11 CLS0:PMODE4,1:PCLS0:SCREEN0,0
:DIM Y(3),B(3),G(3),S(3),P(3),L(
3),V(3):LINE(32,0)-(48,5),PSET,B
F
12 FORX=31TO47STEP4:PSET(X,0,0):
PSET(X+2,1,0):PSET(X+1,4,0):PSET
(X+3,5,0):NEXT
13 FORX=32TO47STEP8:PSET(X,8):PS
```

```
ET(X+4,9):LINE(X,12)-(X+1,12),PS
ET:LINE(X+4,12)-(X+5,12),PSET:LI
NE(X+2,13)-(X+3,13),PSET:LINE(X+
6,13)-(X+7,13),PSET
14 PSET(X,16):PSET(X+1,17):PSET(
X+4,16):PSET(X+5,17):PSET(X+1,20
):PSET(X+5,21):NEXTX:PMODE3:COLO
R2,3:LINE(32,24)-(48,24),PSET:LI
NE(32,25)-(48,25),PRESET
15 PMODE4:GET(32,0)-(47,1),Y,G:G
ET(32,4)-(47,5),B,G:GET(32,8)-(4
7,9),G,G:GET(32,12)-(47,13),S,G:
GET(32,16)-(47,17),P,G:GET(32,20
)-(47,21),L,G:GET(32,24)-(47,25)
,V,G
16 GOTO100:'PAINTING ROUTINES
17 LC=VAL(MID$(PT$,2,3)):TC=VAL(
MID$(PT$,6,3)):RC=VAL(MID$(PT$,1
0,3)):BC=VAL(MID$(PT$,14,3))
18 XX$=LEFT$(PT$,1):IFXX$="Y"THE
N19ELSEIFXX$="B"THEN20ELSEIFXX$=
"G"THEN21ELSEIFXX$="S"THEN22ELSE
IFXX$="P"THEN23ELSEIFXX$="L"THEN
24ELSEIFXX$="V"THEN25ELSERETURN
19 FORYY=TC TO BC STEP2:FORZZ=LC
 TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),Y,OR:NEXTZZ,YY:RETURN
20 FORYY=TC TO BC STEP2:FORZZ=LC
 TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),B,OR:NEXTZZ,YY:RETURN
21 FORYY=TC TO BC STEP2:FORZZ=LC
 TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),G,OR:NEXTZZ,YY:RETURN
22 FORYY=TC TO BC STEP2:FORZZ=LC
 TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),S,OR:NEXTZZ,YY:RETURN
23 FORYY=TC TO BC STEP2:FORZZ=LC
 TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),P,OR:NEXTZZ,YY:RETURN
24 FORYY=TC TO BC STEP2:FORZZ=LC
 TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),L,OR:NEXTZZ,YY:RETURN
25 FORYY=TC TO BC STEP2:FORZZ=LC
 TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),V,OR:NEXTZZ,YY:RETURN
90 'START YOUR PROGRAM HERE
100 PMODE4:PCLS1:SCREEN1,1:PMODE
3
110 PMODE4:COLOR0,0:LINE(0,0)-(2
54,192),PSET,B:LINE(12,8)-(243,1
83),PSET,B:PAINT(2,2),0,0:PMODE3
120 PT$="P000,000-255,008":GOSUB
17:PT$="P000,182-255,192":GOSUB1
7:PT$="P000,000-008,192":GOSUB17
:PT$="P240,000-252,192":GOSUB17
130 GOSUB140:GOTO150
140 DRAW"BM40,120C1ND4R170D4L4G4
D2G2D2G4L2G4L2G4L2G4D6F2R2F2D4L1
16U4E2R2E2U6H4L2H4L2H4L2H2L2H2U2
H2U2H4L4U4":RETURN
150 PAINT(50,122),1,1:PT$="S036,
120-210,146":GOSUB17:PT$="G062,1
48-214,156":GOSUB17:PT$="S048,15
8-210,170":GOSUB17:GOSUB140
160 GOSUB170:GOTO180
170 CIRCLE(60,100),30,1,1.1,.4,.
7:CIRCLE(130,72),80,1,.5,.2,.5:D
RAW"BM160,108C1D4G2D2G4":RETURN
180 PAINT(158,112),1,1:PT$="Y030
,076-160,118":GOSUB17:GOSUB170
190 CIRCLE(120,80),70,1,.5,.2,.5
:CIRCLE(114,72),80,1,.6,.3,.4:CI
RCLE(66,100),26,1,1.1,.4,.6
200 CIRCLE(100,76),30,1,.9:PAINT
(100,58),1,1:PT$="G069,048-120,0
98":GOSUB17:CIRCLE(100,76),30,1,
.9,.6,.2
210 CIRCLE(82,94),30,3,.9,.58,.1
:CIRCLE(130,72),80,3,.5,.3,.47:P
AINT(82,86),3,3:CIRCLE(82,94),30
,1,.9,.58,.1:CIRCLE(130,72),80,1
,.5,.3,.47
220 CIRCLE(194,104),22,1,.9,.33,
.18:PAINT(194,98),1,1:PT$="V168,
084-210,118":GOSUB17:CIRCLE(194,
104),22,1,.9,.33,.18:CIRCLE(160,
120),48,1,1,.6,.88:PAINT(160,100
),2,1
230 CIRCLE(160,120),10,1,1,.75,1
:CIRCLE(160,100),10,1,.9:CIRCLE(
150,90),10,1,.9:CIRCLE(170,86),1
0,1,.9:CIRCLE(140,110),10,1,1,.5
,1:CIRCLE(120,110),10,1,.9,.5,1:
CIRCLE(130,102),10,1,.9,.5,1:CIR
CLE(140,86),10,1,.9,.28,.92
240 CIRCLE(154,82),10,1,.9,.53,.
96:CIRCLE(172,110),8,1,1,.15,.75
:DRAW"BM126,90C1R6F4D2BF8BR4R4BE
20BR6E4BL12BU2U2"
250 COLOR1,1:LINE(16,12)-(239,17
9),PSET,B:LINE(16,146)-(66,146),
PSET:LINE(239,146)-(184,146),PSE
T:PAINT(20,148),3,1
260 CIRCLE(36,10),80,1,.6,.02,.2
5:DRAW"BM36,56C1D10F6"
270 CIRCLE(222,10),80,1,.6,.25,.
49:DRAW"BM222,56C1R6D10G6D74"
280 POKE178,14:PAINT(18,20),,1
290 POKE178,26:PAINT(200,20),,1
300 POKE178,34:PAINT(130,20),,1
310 IFINKEY$<>CHR$(13)THEN310
320 PMODE3:SCREEN1,1
330 IFINKEY$<>CHR$(13)THEN330
340 PMODE3:SCREEN1,0
350 IFINKEY$<>CHR$(13)THEN350
360 PMODE4:SCREEN1,0
370 IFINKEY$<>CHR$(13)THEN370
380 PMODE4:SCREEN1,1:GOTO310
```

The Keyboard Cassette Editing System

By Gary L. Behr

If you've discovered *AUDIOON: MOTORON / AUDIOOFF: MOTOROFF* to audibly search through your cassette tape for a program, you've probably discovered it can be an absolute pain retyping the same commands over and over again. This simple program gives you one keystroke control for *ON/OFF* tape commands, and lets you completely exit the program in a two stroke sequence to begin a new program. It also sounds an audible confirmation for each keystroke. Written with two versions, it functions on either BASIC or Extended BASIC.

The *Keyboard Cassette Editor* program *(KCE)*, used in conjunction with the suggested editing method will allow you to easily update or locate any program on any tape in a matter of seconds. This program is also quite useful for editing out unwanted programs from the middle of tapes and reducing search time to a minimum.

The advantage of listening to a recorded program is it allows a person to identify exactly where and how much space a particular program occupies on the tape. Once this is determined, its position can be noted in an index log using the tape counter for quick retrieval or editing rather than going through the entire tape in the "play-search" mode.

Saving programs from computer to tape isn't hard, neither is loading them back into the computer. What the instruction book doesn't tell you is how to manage and edit your tape once you've begun *CSAVEing* all your programs!

Before discovering *AUDIOON: MOTORON / AUDIOOFF:MOTOR OFF*, I was unplugging the cassette recorder jacks every time I wanted to listen to where a particular program started and finished. When I read the same results could be accomplished using computer keyboard commands

(Gary and his wife Linda own Expressive Image Photography Studio in Madison, Wis. Although Gary has no prior computer background; business needs necessitated the purchase this year of a computer system. The simple programs Gary has written are directed toward efficient computer usage and applications for his business.)

and listening to the tape through the television, life with a computer became a little easier. This also minimized the need to readjust the recorder volume control every time I listened and precluded the possibility of damaging one of the jacks.

Regrettably, not being a speed-typist, constantly re-entering *AUDIOON: MOTORON / AUDIOOFF:MOTOR OFF* became practically as challenging as plugging and unplugging the jacks. There had to be a better way.

> *"Saving programs from computer to tape isn't hard, neither is loading them back into the computer. What the instruction book doesn't tell you is how to manage and edit your tape once you've begun CSAVEing all your programs!"*

I should first explain how I've set up my tape file. Each primary category of interest, such as: business, inventory, data, special programs, home, hobbies, games, etc., has a separate cassette representing "final" *CSAVEs* of each program. I also have a "pending" tape which represents the various programs I'm currently working on but do not yet have in final form.

With other *CSAVE* systems you have a choice of either adding on after the older version of the same program/data or recording over the older version. If you add the "updated" version after the previous version it won't be long before an entire tape has the same program in various stages of updates. If you attempt to record over the older existing version, several problems can arise. You can lose everything with an I/O Error by having a segment of the old version attempt to load with the new version (often caused by less than total erasure of the previous program). Another problem is since the updated version has more information in it, it's also longer and hence uses more tape length when *CSAVEd*, which could record over the beginning of the next *CSAVE* in the tape sequence.

My solution to these potential problems has been very effective for hundreds of error-free *CSAVEs*. First, make sure your recorder is set up properly, including periodically cleaned heads. Second, take full advantage of the recorder's digital counter by zeroing it to the beginning of each tape and setting up a simple log sheet indexing each program to a specific counter value. When you *CSAVE* a program, anticipate the maximum length after all the updates are completed. An effective way to estimate is first to note the length of the original *CSAVE* on the counter. If you think you may end up with twice as much program/data before you're finished, then double the counter length value. After you've determined the maximum length of your program/data then add another five counter units as a buffer between *CSAVEs*.

If I *CSAVEd* a current inventory listing of merchandise and the *CSAVE* began at counter number(CN) 17 and ended at CN 27, I used 10 counter units for the *CSAVE*. Realizing I'm only half done with the current listing I'd have to add another 10 counter units. I may also want to consider if the current inventory will ever be expanded, so to be safe I'd add a 20 percent expansion factor or another four counter units. Now that I have determined the total space needed for a complete inventory, including an expansion factor, my total counter units are 24. Now add five counter units as a buffer between the next *CSAVE* for a grand total of 29. Since my program began on CN 17, I add 29 and when a new (different) program/data is saved on the same tape it will begin at CN 46.

At first this may seem like a lot of blank tape between programs, but compared to sequentially saving updates, it's much more condensed. With an accurate index log you'll be able to fast forward or rewind to the beginning of any program/data in a matter of seconds. And since the program/data always begins at the same place there is no need to revise the index log entries each time a change is made.

Once you have the *KCE* running, your screen display will give you three options: EDIT, STOP or QUIT. Pressing the 'E', 'S' or 'Q' computer key

will initiate each respective function (it's not necessary to use the uppercase SHIFT key). The 'E'dit key serves the same basic function as *AUDIOON: MOTORON*, the 'S'top serves as the *AUDIOOFF:MOTOROFF* and the 'Q'uit allows you to entirely exit the *KCE* program.

It is important to understand the 'Q'uit also erases the *KCE* program, allowing immediate entry of new program information. (This should also be taken into account when initially writing the *KCE* program.) Because of the total exit from the program, when the 'Q'uit key is pressed you are then prompted with "DO YOU REALLY WANT TO QUIT? - YES, NO, BREAK". If you do want to quit just press 'Y', if not, pressing 'N' will re-enter you into the *KCE* program. If you wish to make a temporary "break" from *KCE* without erasing the program itself, just press the keyboard BREAK key and you can review your *CSAVEs'* names by using the BASIC *SKIPF* command. When you're ready to return to the *KCE* editor just enter *RUN*.

You may have noticed the 'E' and 'S' keys are directly adjacent to each other and the 'Q' key is also nearby, which is not sheer coincidence. This position allows easy use for left hand access while the right hand remains free for cassette manipulation, writing notes or whatever. There isn't any need to watch which keys your depressing, as each function key has a beep response with a different tone to audibly confirm every input.

We'll use our previous inventory example to demonstrate the *KCE* program. Let's assume we've already loaded the inventory into the computer from the "final" cassette and updated the file.

CSAVE to "PEND":

CSAVE the updated version on a different, "Pend" cassette which we'll keep just for "pending" actions. Be sure to include at least one backup *CSAVE* (for convenience the *CSAVEs* can be done sequentially).

CLOAD "KCE":

Once satisfied the updated version is safely *CSAVEd*, *CLOAD* the *KCE* program and *RUN* it.

Ready the "Final":

Rewind the "final" cassette to the beginning and "zero" the tape counter. Now fast forward until you're positioned just before the "inventory" program, CN 12-15. Press Play on the cassette recorder and then press 'E'dit on the keyboard. The tape will begin to run and you should hear a faint hum from the blank "buffer" of the previous program as the tape advances. When the tape comes to CN 16 press 'S'top on the keyboard.

"CLEANING" the tape:

Place the recorder in the Record mode and press 'E'dit again. You are now replacing the old inventory program with a clean, blank tape space. Let the recorder "clean" the tape to the end of the full length of the anticipated program (which in this case was CN 46). When you 'S'top, you will not only stop the tape but also place a short audible "beep" on the tape for future editing orientation. (If you 'Q'uit the program while in the 'E'dit mode, the program will automatically turn off the audio and recorder without an orientation

beep recorded on the tape.) Once you've established the tape parameters, subsequent updates will only require cleaning the tape a digit past the end of the previous version.

Position "FINAL" for CSAVE:

Rewind the cassette recorder back to about CN 15, and place the recorder in the Play mode. Press the 'E'dit once more until you are exactly at CN 17 and 'S'top the tape edit.

Reload your program:

Press 'Q'uit and answer the prompt with a 'Y' to totally exit the *KCE* program. Remove the "final" cassette from the recorder and reload your updated inventory into the computer from the "pend" cassette.

Naming your program:

Since you're permitted up to eight characters in your *CSAVE "name"*, a thoughtful suggestion would be to reserve the last three characters for a two-character revision index and separate it with a period or slash such as *"NAME/03"*. This not only allows easy

review, especially during a tape search, but also offers a simple transition to future disk *SAVEs*.

CSAVE to "FINAL":

Once loaded, return to the "final" cassette and *CSAVE* in the normal manner.

Rewind the "final" cassette to the beginning, "zero" the counter, advance to about CN 16 and *CLOAD* the same program you just *CSAVEd*. This reloading will confirm an effective *CSAVE*. (If the "final" *CSAVE* would have failed, you still have several *CSAVEs* on the "pend" cassette to try again.) By zeroing the counter before *CLOADing* the present program, you can determine how much additional tape the update consumed (if the tape now stopped at CN 32, the update portion consumed five additional counter units).

I realize this may sound a bit complex, but most of it is just a logical sequence and won't require much memory. The most obvious advantages are the exceptionally precise control a person has in advancing the tape, either in the Play or Record mode, the audible "beep" to determine the program/ data save boundaries and the ability to main-

tain concise, accurate and consistent records of program/data *CSAVEs*. In the long run it can be a very simple solution to what otherwise might become a complex maze of confusion.

Here's the *KCE* program. It's listed the Extended BASIC version, however by replacing the program subroutine with the alternate subroutine, the program will effectively run on Color BASIC.

For an effective screen display, please adhere to all spacing.

Line 170 lists *NEW;* however while you are initially writing the program replace *NEW* with *END*. This allows you to test *RUN* the program, including 'Q'uit without erasing the entire program. Once you have satisfactorily completed the program, *CSAVE* it for your backup file with the *END* command on Line 170. Then change Line 170 to *NEW* and *CSAVE* it a few times for normal editing use.

You may wish to begin your "pend" cassette with several *KCE* program *CSAVES* for fast access when editing. If you'd like a pre-*CSAVEd KCE* program on a 20-minute cassette with a copy of these usage instructions, please send your name and address along with $8 to 6818 Odana Road, Madison, WI 53719. ◢

Listing 1:

```
10 CLSO:REM"KCE-BY GARY L. BEHR"
20 A$=INKEY$
30 PRINT@100,"   KEYBOARD CASSETT
E EDITOR"
40 PRINT@164,"PRESS"
50 PRINT@196,"<E>  TO EDIT TAPE"
60 PRINT@228,"<S>  TO STOP TAPE"
70 PRINT@260,"<Q>  TO QUIT PROGR
AM"
80 IFA$="E"THENSOUND50,1:AUDIOON
:MOTORON
90 IFA$="S"THENSOUND150,1:MOTORO
FF
100 IFA$="Q"THEN110ELSE20
110 AUDIOOFF:MOTOROFF
120 SOUND200,2:SOUND180,1:SOUND2
00,1:GOSUB180
130 CLSO:PRINT@228,"      EDITING
  COMPLETED":GOSUB260
140 FORT=1TO1000
150 NEXTT
```

```
160 CLS
170 NEW
180 CLSO
190 A$=INKEY$
200 PRINT@100,"DO YOU REALLY WAN
T TO QUIT?"
210 PRINT@215," <Y> YES"
220 PRINT@247," <N> NO"
230 PRINT@279," <BREAK>"
240 IFA$="N"THEN20
250 IFA$="Y"THEN130ELSE190
260 PLAY"P12;T4;V25;L4;G;V18;L8;
C;L8;C;L4;D;L4;C;P4;V22;L4;E;V25
;L3;F
270 RETURN
```

Listing 2:

```
260 FORS=70TO230STEP5
270 SOUNDS,1
280 NEXTS
290 RETURN
```

COMPUTERWARE ®

DON'T FORGET!

Personal Time Management System and Event Recorder with Memos

You won't forget Don't Forget! You'll actually enjoy getting organized with this personal time management systems. The Macintosh-like icons make entering your personal schedule simple and fun. But if you need instructions, there are help screens to assist you.

You'll never miss a birthday or important appointment again! With Don't Forget! you can record the entire year's occasions and daily appointments ahead. Each day has spaces for 4 Special Occasions, 2 Memos, and hourly notes for 6 am through 9 pm. The built-in 51 x 24 upper and lower case hi-res display makes it very easy to read.

You can display or print any daily schedule – or a whole week at a time – so you'll remember every important event. You can even print a blank monthly calendar page with big boxes to scribble notes in!

So use your CoCo's memory and Don't Forget! as your personal secretary. Designed for mouse, joystick, or keyboard entry. Don't Forget! requires 32K and one disk drive.   **Disk $27.95**

PUT A MONITOR ON YOUR COCO WITH VIDEO PLUS!

with audio!

| NAP Green Screen | $109.95 | NAP amber screen | $114.95 | Color with audio | $275.00 |

NEW! IMPROVED! HIGHER RESOLUTION

We proudly introduce a **new** model of our popular NAP monochrome monitors with **higher resolution** and a more **compact** stylized case. Of course they will have the **built-in audio** – no need to add expensive amplifier or go without sound.

Yes! You can enjoy the crisp display of a composite video monitor using **Computerware's Video Plus** interface. Each is fully **assembled** and **tested**. Installation is quick, easy, and requires **no soldering**. Your TV output is not disabled. **Audio output** available. Choose the model right for your computer and monitor.

VIDEO PLUS	**$24.95**

Interfaces the **original** model of Color Computer to **any** composite video monitor (color or monochrome).

VIDEO PLUS II M	**$26.95**

Interfaces the **CoCo II** with a **monochrome** composite video monitor.

Video Plus II C	**$34.95**

Interfaces the **CoCo II** with **any** composite video monitor (color or monochrome).

VIDEO PLUS II U	**$34.95**

Interfaces the new CoCo II (model with power switch next to the TV output and chips soldered, not socketed) with any composite video monitor (color or monochrome).

ICE HOCKEY

Begin with our national anthem, then race into action in hi-res graphics with realistic player & puck movement. One or two players compete against the real-time clock. Many skill levels. (Requires 32K)

cass $24.95   disk $27.95

DISK SYSTEMS

The biggest investment you'll make in your computer system will be a disk system. It will widen your computing horizons considerably and should be chosen carefully. Computerware configures systems to include the most **reliable** components at the best prices available. We do not try to be the cheapest by compromising quality!

We offer **complete systems** assembled and tested by **expert technicians**. We've been selling disk systems since 1977 so we know what you need and we make sure you get it! We use the highest quality components including **TEAC** and **Hitachi** drives. **J & M Systems** controller, all **GOLD** connectors, **heavy duty** cabinet, power supply, and cable. Our power supplies are totally adequate to insure reliability; we know dual drives need heavier power supplies! All of our TEAC and Hitachi drives come with a **6ms** step rate, full **40 tracks, slim** line size, and a one year warranty. In addition to the DOS manual, you receive a 20-page **technical manual** documenting every detail of disk drive performance and specification.

Also available from Computerware is the new Amdisk. Compatible with Radio Shack Drive 0, it is perfect for adding extra storage. Our software is available on Amdisk format for only $4 extra. (Remember every Amdisk cartridge is equivalent to two floppy diskettes.)

DISK SYSTEMS

Include J & M Controller, DOS Manual, cabinet, power supply, half-size double density drive, & all cables. (Prices quoted for JDOS. Add $10 for RSDOS.)

Single drive, Single sided	$349.00
Single drive, Double sided	$399.00
Dual drives, Single sided	$509.00
Dual drives, Double sided	$599.00

ADD ON DRIVES

Each drive add-on includes half-sized drive(s) in cabinet with extender.

Single drive, Single sided	$229.00	J & M Controller	
Single drive, Double sided	$269.00	with JDOS	$130.00
Dual drives, Single sided	$399.00	with RSDOS	$140.00
Dual drives, Double sided	$469.00	JDOS rom and manual	$ 39.95

AMDISK III

Complete Dual Drive System including controller, cable & manual. (Price quoted for JDOS. Add $10 for RSDOS.)	$499.00
Dual Drive and cable only.	$375.00

DISKETTES

Computerware double density diskettes with hub rings are of the highest quality. We love them and so will you!

5-pack **$10.95**	10-pack **$19.95**	amdisk cartridges **$5.50** each

CoCo Cookbook

• Can be used for ANY free-format filing system (not just recipes). Try it for periodicals and article synopsis, product reviews, real estate descriptions....!
• Store & retrieve a large number of recipes.
• Up to 270 recipes on a single disk using a special compression technique.
• Up to 3040 characters per recipe including title, ingredients, & instructions – all in easy to use "free form" format.
• Access each recipe by title, number, or with special keyword search (like all the recipes using "chicken!") List on the screen or printer.
• 50 recipes included FREE!
• Requires 32K and a disk drive.

32K disk **$27.95**

SAM SLEUTH

As the detective for Sam Sleuth Investigations, you are given 3 cases   (of increasing difficulty):

Case of the Missing Cat
Mystery at the Museum
Baffling Bank Robbery

Using your sleuth skills (& joystick or mouse), you search the town (displayed in hi-res graphics), interviewing townspeople, collecting evidence, examining the grounds, & gathering data to solve the mysteries & apprehend the culprits. Remember, you are working against the clock!

Investigate the bank, museum, school, offices & homes, market, gas station,...Talk to Mike, Willy, Roy, Sue, Chrissy,... And keep track of those addresses & clues!

The unique graphics presentation & mouse/joystick control are as intriguing as the mysteries! (Requires 64K & mouse or joystick)   **cass $24.95   disk $27.95**

MAJOR ISTAR

Under the Doomed Sea
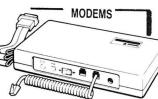
SCENARIO: You travel to TRIDENT RESEARCH DOME because an urgent call for help is received from one of the service droids stationed there. He said help was urgently needed, but before he could say why, his transmission was cut off!

OBJECT: Solve the mystery at TRIDENT in as few moves as possible.

SETTING: In the beginning of the 21st century, undersea cities, interstellar spaceships, colonies in other planets & solar systems, worker droids, super computers, are all realities. One hero of the time is MAJOR ISTAR. In the late 1990's when space exploration & colonies began, it was necessary to form a task force to offer help to all that needed it in these hostile environments. You, Camerion J. Istar, are a highly rated member of that team and you have never failed to solve a mystery!   **cass $24.95   disk $27.95**

Spread your wings, child, and fly. Glide over the Plains of Content. Use your eagle eyes to find the Mountains of Intuition. Soar on their uplifting currents. Trust your psychic tailwind.

— Laran Stardrake

Binary Birthday Cakes, Letters And More Wordsworths

By Bob Albrecht and Ramon Zamora
Rainbow Contributing Editors

Basic Skills

Basic Skills is a book by Herbert Kohl. It is a wonderful and thoughtful book — a book by a philosopher and doer who has spent his life learning from children about children. We wish that every person concerned about the growth and well-being of children would read this book.

From the introduction:

I believe in public education but do not believe that the public schools are providing an adequate basic education for our children. It is important to emphasize this distinction between public education and the current state of the public schools, between what could be and what is. We can make a decent system of public education and help our children learn to take control of their lives rather than be managed, controlled, bossed, and bewildered. This book presents a concrete program for the revitalization of public education. It is based on the idea that the fundamental goals of public education should be to develop informed, thoughtful, and sensitive citizens who are able to fulfill their personal needs at the same time that they contribute to making the whole society harmonious and compassionate. We need this wholeness now more than at any time I can remember. It is

"School is in the Heart of a Child" is for parents of quite young children. We want to help you work and play with your three- to eight-year-old child and learn to use computers as a joyful family experience. We want to suggest ways to incorporate the home computer as another means to encourage your child's independence, growth, and control over his own life. See the pride on her face as she directs the computer to do what she with deliberation selects. See her head gears switch to "on" as she progresses step-by-step with your presence and caring direction.

We will explore (we hope, with *your* help) the following:

- Specific "teaching" techniques so that the discovery can be the child's own.
- Critical evaluation of software based on extensive playtesting in family and related enviroments.

- Additional resources to consult: books, magazines, software publishers, networks, etc.
- Suggestions for interludes and fun times away from the computer (a must): call the librarian for specific information; watch a TV program together and discuss it; work together as volunteers in a community project; take a spring (or fall or winter or summer) awareness walk . . .
- Whatever we learn from families we work with in Menlo Park or from you, our readers. Let's pool our knowledge. Let's share our experiences as we all learn from our children.

We also provide small programs you can type in and use right now.

Copyright© 1984 by DragonQuest, P.O. Box 310, Menlo Park, CA 94026.

(Well-known author Bob Albrecht also writes the "GameMaster's Apprentice" feature for The Rainbow each month. Ramon Zamora is author and co-author of several books, co-founder of Computer Town USA!, and currently designing computer games for kids at Child Ware Corp. in Menlo Park, Calif.)

common to hear people talking about the United States suffering from "too much democracy," "an excess of freedom" and "unbridled liberty." It is time to take a serious look at ourselves and see how much democracy we really have and how far we have to go to achieve the dreams expressed by the Declaration of Independence, which dared to call life, liberty, and the pursuit of happiness rights and not privileges.

The focus of this book is on public education and the struggle to create decent schools for all of our children. I hope it will move people to stay engaged in that effort and to develop concrete programs for action in their own communities.

We suggest you get this book. If your local bookstore doesn't have it, order it directly from the publisher: *Basic Skills* by Herbert Kohl, from Bantam Books, Inc., 414 East Golf Road, Des Plaines, IL 60016, $3.95 plus $1.25 postage and handling.

Binary Birthday Cake

Remember your first birthday? Maybe the cake looked like this:

Well, on your second bithday there were two candles, on your third birthday three candles, and so on. As we grow older, the number of candles gets embarrassingly large and harder to blow out.

Sooo . . . why not try a binary birthday cake?

ONE YEAR OLD

TWO YEARS OLD

THREE YEARS OLD

FOUR YEARS OLD

FIVE

SIX

SEVEN

With four candles, we can go up to 15 years old, with five candles, up to 31 years old and with six candles, up to 63.

But if you are 57, get people to look at the cake from the back instead of the front . . . in fact, if you don't put something on the cake to indicate the front, your age will be ambiguous (unless, of course, it is a binary *palindrome!*). And, for all you computer people out there —

FRONT: 57

BACK: 39

A PALINDROME

How many candles would Methuselah need?
How many candles would Gandolf need?
How many candles would a dragon need?

Here is a cake with an extra BYTE

RAINBOWfest

Irvine

Feb. 15-17

Show Schedule:

Friday evening — Exhibit hall open from 7 p.m. to 10 p.m.
Saturday — Breakfast at 8 a.m. Exhibit Hall opens at 10 a.m. and closes at 6 p.m.
Sunday — Exhibit Hall open from 11 a.m. to 4 p.m.

Joining in the fun and excitement of **RAINBOWfest** is a great way to get to know the CoCo Community. Many of those who write for THE RAINBOW —and those who are written about — attend CoCo's very own show. It's a people-to-people event as well as a valuable learning experience.

For the 1984-85 season, we've scheduled three **RAINBOWfests** in three parts of the country. If you missed the **RAINBOWfest** in Princeton, N.J., why don't you make plans now to be with us in Irvine, Calif., or Chicago, Ill.? Each show will offer fun, excitement, new products, seminars and information for your CoCo! And for those who (perish the thought) don't like CoCo as

much as you, we've scheduled each **RAINBOWfest** in an area that will provide fun and enjoyment for the whole family.

Our Irvine, California, show is being held at the Irvine Marriott Hotel, which offers special rates for **RAINBOWfest**. The show opens Friday evening with a 7 p.m. to 10 p.m. session. It's a daytime-only show Saturday — the CoCo Community Breakfast is at 8 a.m., then the exhibit hall opens promptly at 10 a.m. and runs continuously until 6 p.m. There will be no exhibition hours or seminars Saturday evening. On Sunday, the exhibit hall opens at 11 a.m. and closes at 4 p.m.

Our highly popular CoCo Community Breakfast will again feature a well-known figure from the Color Computer Community. And the exhibition will be interspaced

with a number of seminar sessions on all aspects of CoCo — from writing in machine language to making your BASIC work better.

But most of all, there will be exhibitors. Lots of them. All ready to demonstrate products of every kind. Some with special programs and hardware items to introduce. Others with show specials.

Tickets can be secured directly from THE RAINBOW. We'll also send you a special reservation form so you can get your special room rate.

Come to **RAINBOWfest** . . . help us all celebrate CoCo Community at its finest.

United Airlines and THE RAINBOW have joined together to offer a special discounted fare to those attending **RAINBOWfest**-Irvine. Simply by calling United at the toll free number listed below and identifying our meeting, with account number 522-I, you will be eligible for a 20 percent discount on the Easy Saver Fare. The only requirement is a Saturday night stay.

RAINBOWfest-Irvine, California (L.A. area)
Date: February 15-17, 1985
Hotel: Irvine Marriott Hotel
Rooms: $65 per night, single or double
Advance Ticket Deadline: February 8, 1985

RAINBOWfest-Chicago, Illinois
Date: May 17-19, 1985
Hotel: Hyatt Regency Woodfield
Rooms: $49 per night, single or double
Advance Ticket Deadline: May 10, 1985

(800) 521-4041
Account Number 522-I

RAINBOWfest Irvine
Seminar Program And Speakers

• Linda Nielsen

The CoCo Artist
High Res Graphics

Linda, of Moreton Bay Laboratory, and several others active in the CoCo area, will demonstrate some graphics and help you to learn about some techniques you can use.

• Dale Puckett

Beginner's Tour Of OS-9
Beginner's Tour Of BASIC09

A free-lance writer and programmer, Dale has worked with microprocessors since 1976, and is the author of *The Official BASIC09 Tour Guide*. Dale will be available to sign copies of his new book, *The Complete RAINBOW Guide to OS-9*.

• Bob Albrecht

School Is In The Heart Of The Child

Bob Albrecht, RAINBOW columnist who writes "School Is In The Heart Of The Child," is one of the most prolific authors in the microcomputer world today. A specialist in writing for beginners, he is the author of numerous books, including *TRS-80 Color BASIC*.

• Jim Reed

Writing For RAINBOW

Jim, managing editor of THE RAINBOW, will talk about how you can submit programs and articles to magazines for fun and profit. He is also senior editor of PCM — and editorial director of SOFT SECTOR (for the Sanyo).

• Dan Downard

Inside Your Color Computer
Software And Hardware Interfacing

Dan Downard is the technical editor for THE RAINBOW and an electrical engineer. He has been involved in electronics for 24 years through ham radio (K4KWT). His interest in computers began about five years ago and he has built several 68XX systems.

• Don Inman
• Tim Finger
• Bob Albrecht
Along with several other panelists

A Realizable Fantasy:
The Home Dream Machine

Don Inman is a co-author of a series of booklets for Radio Shack titled *Color LOGO Guide for Teachers*. He is a former teacher and is presently a full-time author with the DYMAX group.

Tim Finger is a member of the DYMAX group, along with Don and Bob.

• Wayne Day

Exploring CompuServe
and the Color SIG

The CoCo User and the
Local Bulletin Board

Wayne Day, a traffic engineering signal technician, is the SYSOP of The Color SIG of CompuServe, the world's largest consumer information service. He is also a certified paramedic and works part-time for an Emergency Medical Service provider. His amateur radio operator call sign is WA5WDB.

• Bill Nolan
• Bob Albrecht
GameMaster's Apprentice And
Dragon's Byte

Bill Nolan, who teaches "Programming In BASIC" at the college level, owns Prickly-Pear Software Co. and has written several commercially successful software packages.

•Ken Kaplan, president of Microware Systems Corp., will be our keynote speaker at the CoCo Community Breakfast, Saturday morning at 8 a.m.

Adventure Games For The Rest Of Us

We are looking, mostly in vain, for *easy-to-play* Adventure games rated G. We are tired of games that depend on killing as the way to success. Last time we recommended *Adventure Starter* from Owls Nest Software, P.O. Box 579, Ooltewah, TN 37363. For 16K Extended Color BASIC. On cassette for $17.95 postpaid.

We encourage all of you who want to try an easy Adventure game with your kids to get this Adventure and do it. Please share your experiences with us. We will continue playing *MYHOUSE* with kids and share our experiences with you. Why do we choose *MY-HOUSE*? Because it is the only non-violent, easy-to-play Adventure game we know of for a 16K CoCo with Extended Color BASIC. As you play *MYHOUSE*, encourage your kids to make a map. Please send us maps of *MYHOUSE* and tell us about the process of learning how to make the map. Next time, we will share with you some of our map making methods.

WANTED: Non-violent, easy-to-play Adventure games. We especially want games suitable for parent and young child to play together. Why do all you Adventure game designers spend your time making increasingly more difficult games for the "elite" and "sophisticated" player? The world is full of beginners. Why not make beautiful games for them? Instead of selling 1,000 incredibly complex games to the Adventure game cult, why not sell 100,000 beautiful, non-violent games to beginners, games for "the rest of us."

We are also looking for non-violent Adventure games with several levels of play. A beginner starts with an easy game and is guaranteed moderate success, then moves up to a more difficult level, and so on. Focus on exploration and problem-solving instead of "kill monster, get treasure." Make your games realizeable realistic fantasies. As Laran Stardrake once said, "Reality expands to fill the available fantasies."

We Love The Letters!

Our greatest pleasure in writing this column is hearing from you, our esteemed RAINBOW readers. Your letters guide us as we evolve "School is in the Heart of a Child."

Thanks, Carl H. Bloss, for two letters. We have combined them as follows:

As a new subscriber and avid reader of THE RAINBOW, I feel you should also readdress your column "School is in the Heart of a Child" to teachers, librarians and computer aides in the school. Little is done to help these people at the early critical years of children with computers. I know, I am a teacher, librarian and erstwhile programmer at a 5K level.

As an educator, I am fully aware that written curriculums often do not allow enough freedom for both the teacher and the student to explore — to work and play while learning. Schools approach learning from such a "stuffy" position that everything must be justifiable, accountable through criterion-referenced testing, or everything must have a planned outcome.

There are two schools of thought with LOGO, one following an experimental approach, the other using a planned teaching course of study. Most schools seem to be using the more restrictive planned teaching approach, even at the

earliest levels. This is contrary to all of Papert's concepts as well as most learning theories.

Our school will have a lab of Color Computers, but not located in the library media center where I work. Since librarians work closely with the teaching staff, we are often asked for sources with ideas and materials on a variety of computer subjects. Your column, re-addressed to encompass the teacher and librarian, is a tremendous source of ideas.

Thank you very much, Carl. If I ran your school, I would give you a bunch of CoCos that kids could check out and take home overnight. People like you can blend the worlds of learning — in school and out of school — so children can have the best of both worlds. We'll send you a stuffed dragon. Put him or her in your library media center to remind you to keep plugging away and write to us now and then.

We are getting lots of letters! Thanks. Our next letter is from June Lemons.

I applaud your column and your concept about the education of children. (I strongly suspect many parents are also losing their hesitancy to try using their computers by trying your programs.)

My husband and I entered the fascinating, educational, fun world of computing a short year ago. The thrill of learning all the marvelous things our CoCo can do continues on. I contrive to spend as much time as possible reading — and keying in — articles and programs from THE RAINBOW, cheerfully rising early and retiring late in my pursuit of knowledge, participating via others' creativity.

About three months after getting our CoCo, I wrote an extremely simple program for my two-year-old grandson, a program he can use to learn: 1) letter keys; 2) how to spell his name (of course, the program is easily adaptable for any child), rewarding him with his (then) favorite music, "Happy Birthday."

I have two regrets: getting acquainted with computers so late in life, and that my grandson's parents don't share my enthusiasm. But when Bennie visits, he asks to go into what he calls my computing room where he displays a marvelous grasp of letters, colors, shapes — after all, he knows what unicorns and pedometers are!

My husband has had one disappointment — I haven't been able to "draw" a cake with icing and a flickering candle to complete "Bennie's program."

Thanks, June. We think you might like "Binary Birthday Cake" in this episode of "School is in the Heart of a Child." We'll ask Don Inman how to draw the cake with flickering candles. Of course, we will send you a stuffed dragon to play with Bennie, you and the CoCo.

Letters are close to the heart of this series. We are getting lots of letters, thank you, and want more! Our next letter is from John A. McGoldrick. John has a larger sample size than most of us — he has five children, ranging in ages from three to nine.

I have just finished reading your

```
100 REM**WORDSWORTH 1&2 SCH 10-1          500 REM**COMPUTE W1 & W2
110 CLS                                   510 FOR L=1 TO WL
199 '                                     520 : L$ = MID$(WRD$,L,1)
200 REM**ASK FOR A WORD                   530 : IF L$<"A" THEN 580
210 PRINT                                 540 : IF L$>"Z" THEN 580
220 INPUT "YOUR WORD"; WRD$               550 : LS = ASC(L$) - 64
299 '                                     560 : W1 = W1 + LS
300 REM**WL IS LENGTH OF WORD             570 : W2 = W2*LS
310 WL = LEN(WRD$)                        580 NEXT L
399 '                                     599 '
400 REM**INITIALIZE W1 & W2               600 REM**PRINT BOTH WORDSWORTHS
410 W1 = 0                                610 PRINT "WORDSWORTH #1 IS" W1
420 W2 = 1                                620 PRINT "WORDSWORTH #2 IS" W2
499 '                                     699 '
                                          700 REM**GO FOR ANOTHER WORD
                                          710 GOTO 210
```

article in THE RAINBOW and was quite impressed with it. Educational software was the overwhelming reason for the purchase of my Radio Shack 64K ECB this past January. I think your department "School is in the Heart of a Child" will help to fill the void that now exists.

I am the father of five children, ages nine, eight, six, five, and three. I cannot afford to buy software either based on advertisements or solely on the manufacturer's name — I have been burned both ways . . .

Parents want good educational software. They want software that will help their children become computer literate, but will not cause parents to become financial paupers in this quest.

Special thanks, John. I hope the CoCo "cottage industry" hears you. We, too, are tired of software that simply exploits the desire of parents to use the CoCo to help kids learn.

Yes, software is too expensive. You people out there who have kids, what do you want? What will you pay $9.95 for? $14.95? 19.95? and so on. What do *you* want?

If you have young children, compare the "holding power" of software with *Sesame Street.* Marido (age six), Bob, and Ramon will gladly watch *Sesame Street* repeats, several times — and that's free! Software should also have "holding power" and grow with the child. Home software should have vertical growth; school software is different — it is designed to be horizontal — more about this next time.

Wordsworth

Hooray! Several of you sent answers to our Wordsworth challenge in the August issue. Here again are the questions you answered.

1) What magic word has a Wordsworth #1 equal to the number of weeks in a year?

2) What five-letter word has the smallest Wordsworth #2?

3) What five-letter word has the largest Wordsworth #2?

4) What is the longest word (most letters) that has a Wordsworth #2 less than 20,000?

We have received answers from Amos and Josh Goldie, Lara Hansen, the Jones children, Stephen Lathom, Eric Lauterbach, and Andrea and Jenny McGinnuess. Next time we'll share their answers with you. In the meantime, send in *your* answers. Here is a Wordsworth program to compute both Wordsworth #1 and Wordsworth #2.

To answer a Wordsworth question, browse a dictionary for words to try with the above Wordsworth program. Let's all use *The American Heritage Dictionary of the English Language*, paperback edition, published by Dell Publishing Company, 1 Dag Hammarskjold Plaza, New York, NY 10017. You can buy it for $3.95 at almost any bookstore or order it from the publisher.

Here are more questions for you Wordsworth fans. Remember, all these questions refer to the *American Heritage Dictionary* mentioned above.

5) In the entire dictionary, what word has the largest Wordsworth #1?

6) In the entire dictionary, what word has the largest Wordsworth #2?

7) What is the first word (alphabetically) to have a Wordsworth #1 of exactly 100?

8) In the entire dictionary, what word has a Wordsworth #2 closest to 1,000,000?

Send us your answers. We will put all answers received by December 31, 1984 in a box, shake 'em up, and draw a winner. The winner will receive, of course, a stuffed dragon.

DragonSmoke

Koala Pad is alive and well at Radio Shack! Look for it on Page 164 of the 1985 Radio Shack Catalog No. 380 or Page 52 of the 1985 TRS-80 Catalog No. RSC-12. They call it the TRS-80 Touch Pad, Catalog No. 26-1185, price $59.95. OK, cottage industry — let's get busy and write software for this marvelous addition to the CoCo family.

We have decided to publish a little newsletter called *DragonSmoke* about four times a year. If you want a free copy, send a self-addressed, stamped envelope to *DragonSmoke*, P.O. Box 310, Menlo Park, CA 94026.

16K
ECB

Helping Youngsters Form Categories

By Steve Blyn
Rainbow Contributing Editor

The young child deals with new items and experiences as unique events. He has a small fund of knowledge to draw from. He has yet to learn to classify things into groups of similar things. Children often seem fascinated by what appears to us to be something quite commonplace; they cannot place the item into a familiar category.

As we grow in experience, we begin to define items into narrower categories. The Color Computer can be an example — a beginner might at first think of the CoCo as merely a type of computer. The more experienced user knows it is in the 6800 family of computers rather than the 6500 or Z80 families. These distinctions become more important as your knowledge and needs increase.

Software is another example. Upon purchasing a computer, most people usually want to accumulate almost any kind of software. There is a need to see what the new machine can actually do. After a while, we break the software down into categories or specific areas such as business, education, leisure, etc.

(Steve Blyn teaches both exceptional and gifted children, holds two master's degrees and has won awards for the design of programs to aid the handicapped. He and his wife, Cheryl, own Computer Island.)

We begin to specialize. Many of us then further specialize within an area to several favorite specific companies. Some of us even go further into specific requirements within the programs of specific companies such as OS-9 programs only or networkable programs only.

This type of refinement of categories is part of the human experience. We constantly redefine as our knowledge

> *"Too often, school curricula require some of the children to deal with abstract concepts before they have secured sufficient experience on earlier levels of experiences in that subject. This can lead to inadequate and insecure learning."*

grows in any field. This helps us to deal with our world. There are too many items and events to deal with individually. As we grow, we learn to place them in various categories to help us effectively deal with the world around us.

Learning proceeds from concrete to abstract concepts. First we can deal only with single objects and later with

categories. This process proceeds according to age and experiences at individual levels. Too often, school curricula require some of the children to deal with abstract concepts before they have secured sufficient experience on earlier levels of experiences in that subject. This can lead to inadequate and insecure learning.

The famous child psychologist, Jean Piaget, once performed an experiment to help study the levels of categorization in children. Three girls, aged seven, nine and 11 were asked to pack clothing in two suitcases. The way they would decide to arrange their clothing into the suitcases could reveal much about their thought processes.

The seven-year-old first filled one suitcase randomly with clothes until it was full and then packed the other with the remaining clothing. No real thought was given to any categories of clothing.

The nine-year-old thought for a while, and then packed all clothing worn above the waist in one suitcase and clothing worn below the waist in the other. The top part of her pajamas and a two-piece bathing suit were placed in one suitcase and the bottoms in the other.

The 11-year-old was more adult-like in her thinking and packed clothes worn during the day in one suitcase and night clothes in the other. It can easily be

seen from this experiment different levels of organization of the same items.

This month's program naturally deals with grouping and categorization. The key element in this program is the *DATA*. We had to select one area and chose sports terminology. This was purely subjective on our part: one subject had to be chosen. Please do not restrict your program to our chosen topic.

The *DATA* that you choose to enter should be age appropriate for your child. Younger children might, for example, group food items by fruit, vegetable or dairy. Older students might group chemical compounds by acid, base or salt. The *DATA* that you enter could also be the child's choice. This would further involve the student in the computer.

The goal for the user of this program is to arrange the 12 words properly underneath the three category words. The words are moved one at a time by the arrow keys. After the twelfth word is moved, a list of the 12 words in their correct categories appears. The child can compare this list to the one that he has just completed.

Lines 40-50 set the dimensions and read the *DATA*. Line 70 prints the data words. These are the A$ words: The B$ letters will never be printed on the screen. Line 80 prints the category headings. (Your headings may vary from ours). Line 140 prints one of the words and lines 120-130 set the limits of the word so that it does not mess up the screen.

Lines 180-250 let you move the word around the screen. The arrow keys are represented by the character string (*CHR$*) numbers 8, 9, 10 and 94. When one of the arrows is pressed, the key word to be classified is moved. *CHR$(13)* represents the ENTER key. When it is pressed, the next word appears. Lines 300-360 read and print out the correct list for comparison purposes.

The *DATA* on Line 410 consists of 12 pairs of information. A word to be classified and an abbreviation for the classification make up one pair. They are the A$ and B$ portions. The abbreviations do not get used until needed on lines 320-340. Here, the computer interprets the B$ part and places the word in the correct category.

The *DATA* supplied is merely one of literally thousands of possible choices. Use your imagination to help your children better classify things or events in their schoolwork and daily lives.

190 190
END 218

The listing:

```
10 REM"CATEGORIES"
20 REM"STEVE BLYN,COMPUTER ISLAN
D,1984
30 CLS0
40 DIMA$(12),B$(12)
50 FORT=1TO12:READA$(T),B$(T)::N
EXT
60 PRINT"        here is the word l
ist"
70 FORT=1TO12:PRINTA$(T),:NEXTT
80 PRINT"   baseball    tennis    h
ockey"
90 B$=STRING$(8,128)
100 N=451:Y=1
110 REM"SET LIMITS OF WORD PLACE
MENT"
120 IF N<259 THEN N=259
130 IF N>476 THEN N=476
140 PRINT@N,A$(Y);
150 J=258:FORT=1TO4:PRINT@J,CHR$
(249);:J=J+32:NEXTT
160 K=268:FORT=1TO4:PRINT@K,CHR$
(249);:K=K+32:NEXT T
170 L=277:FORT=1 TO 4:PRINT@L,CH
R$(249);:L=L+32:NEXT T
180 REM"USE THE ARROW KEYS TO MO
VE THE WORDS"
190 A$=INKEY$
200 IF A$=CHR$(8) THEN PRINT@N,B
$;:N=N-1:GOTO120
210 IF A$=CHR$(9) THEN PRINT@N,B
$;:N=N+1:GOTO120
220 IF A$=CHR$(94)THEN PRINT@N,B
$;:N=N-32:GOTO120
230 IF A$=CHR$(10) THEN PRINT@N,
B$;:N=N+32:GOTO120
240 IF A$=CHR$(13) THEN Y=Y+1:N=
451:SOUND150,2:IF Y>12 THEN 270
ELSE 120
250 GOTO 190
260 REM"GET & PRINT THE CORRECT
LISTS"
270 PRINT@384,STRING$(32,239);:F
ORT=1TO9:PRINTSTRING$(32,128);:N
EXTT
280 PRINT@194,"here are the corr
ect answers";
290 D=227:E=237:F=246
300 RESTORE:FOR G=1TO12:READA$(G
),B$(G)
310 FORG=1TO12
320 IF B$(G)="B" THEN PRINT@D,A$
(G);:D=D+64
330 IF B$(G)="T" THEN PRINT@E,A$
(G);:E=E+64
340 IF B$(G)="H" THEN PRINT@F,A$
(G);:F=F+64
350 SOUND230,1
360 NEXT G
370 PRINT@486,"PRESS ENTER TO GO
 ON";
380 EN$=INKEY$
390 IF EN$=CHR$(13) THEN RUN
400 GOTO 380
410 DATA LOVE,T,ICING,H,FACEOFF,
H,TRIPLE,B,DUECE,T,GOALIE,H,MOUN
D,B,RACKET,T,PUCK,H,PLATE,B,BALK
,B,FAULT,T
```

TAPE UTILITY

*Here's a program that concatenates RAM
records and writes them on tape. It's your miss-
ing . . .*

By H. Allen Curtis

Can you do the following with a
single cassette load command?

1) Load the text screen to display an
introductory message or low resolution
picture;

2) Load a BASIC program;

3) Load the graphics screen with a
high resolution picture for subsequent
display;

4) Load automatically memory pro-
tected high RAM with assembly lan-
guage routines to be called by *USR*
functions;

5) Automatically start the BASIC pro-
gram; and

6) Provide some piracy protection
for your program.

If you cannot, then you are missing
Link. No, I did not mean that you are
the missing link, but that you are miss-
ing out by not using the *Link* program
to be presented in this article.

---

*(H. Allen Curtis resides in Williamsburg,
Va. He is interested in 17th and 18th
century history and enjoys biking
through the colonial capital. He balances
past and present with his computer
work.)*

Link is not a pre-loader. That is, *Link*
does not have to be loaded into RAM
before you issue the command to load
your program, screens and subroutines.
Link concatenates (links) as many as 10
non-contiguous RAM records and
writes them on tape. A record is defined
here as any program (BASIC or assembly
language), any contiguous assembly
language routines, or any set of stored
data. The linked records written on tape
are simply loaded by means of BASIC's
CLOADM command.

If you want to employ *Link* to record
and auto-start an assembly language
program instead of a BASIC program,
you can readily do so. In fact, *Link* is an
assembly language program which will
be used to record itself.

Link has the ability to write a record
from one RAM location and load it into
another specified RAM location with-
out recourse to the offset feature of the
CLOADM command. Thus, for in-
stance, you may design several text
screens, transfer them to new locations
in RAM and then use *Link* to write
them on tape for future sequential load-
ing and display on the text screen. The
fact that you relocate one or more
records such as screens does not require
you to relocate the other records to be
linked and written.

The order in which records are con-
catenated is left to your discretion. If
you, for example, have more than one
text screen to be loaded and displayed,
you would probably load one or two
records between screen records to allow
the screen to be displayed for a suffi-
ciently long time.

Link can be employed as part of a
protection process for your programs.
How *Link* can be used in this way will be
discussed in detail at the close of the
article.

The program of Listing 1 generates
Link and stores it in RAM. The strings
in lines 20 through 120 of Listing 1 are
messages used by *Link* to prompt you in
the process of concatenating records
and writing them on tape. The values in
the *DATA* statements of lines 210
through 520 comprise *Link* routines
that actually do the linking and writing
of the records on tape. The values in the
remaining *DATA* statements form the
major portion of the first record to be
written on every *Link* produced tape.
You do not specify this record. The
record is *Link's* means of altering the
usual *CLOADM* sequence of instruc-
tions to permit the proper loading of
concatenated records.
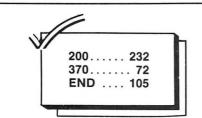
Incorporated into Listing 1 is a check
on the accuracy of your typing of the

DATA statements. Thus, with the use of Rainbow Check Plus you are doubly aided in the correct typing of Listing 1. When you have correctly typed Listing 1 and run the program without the occurrence of any error messages, save it on tape: Type *CSAVE"GENLINK"* and press ENTER.

After saving the program of Listing 1, run it again. Then type *EXEC* and press ENTER. This action will produce the first prompt of *Link*. *Link* requires you to provide a filename for the concatenated records to be put on tape. Usually the filename will be that of the main pro-

```
200 ...... 232
370 ....... 72
END .... 105
```

Listing 1:

```
10 'ENTRY ADDRESS IS &H1100; FIR
ST ADDRESS IS &H1000; LAST ADDRE
SS IS &H132B
20 A$="TYPE & ENTER
30 B$="FILENAME:
40 C$="TYPE (IN HEX)
50 D$="ENTRY ADDRESS:
60 E$="POSITION TAPE
70 F$="FIRST SOURCE
80 G$="LAST
90 H$="FIRST DESTINATION
100 I$="ALL RECORDS SPECIFIED?
 (Y/N)
110 J$="READY CASSETTE TO RECORD
120 K$="THEN PRESS ENTER
130 X=256*PEEK(VARPTR(A$)+2)+PEE
K(VARPTR(A$)+3)
140 FORI=0TO 174
150 POKEI+&H1000,PEEK(I+X)
160 IFPEEK(I+X)=0 THENX=X+8
170 NEXT: IFPEEK(398)=57THENPOKE3
99,174:POKE400,64:POKE398,126
180 FORI=0TO596:READL$:L=VAL("&H
"+L$):E=E+L:POKEI+&H10AF,L:NEXT
190 FORI=0TO76:READL$:L=VAL("&H"
+L$):E=E+L:POKEI+&H1E2,L:NEXT
200 IFE<>78082 THENCLS:PRINT"DAT
A ERROR"ELSEPOKE&H9D,17:POKE&H9E
,0
210 DATA 5F,30,1,A6,84,26,FA,8D,
1F,84,F,97,7D,8D,12,9A,7D,97,7D
220 DATA 8D,13,84,F,97,7C,8D,6,9
A,7C,97,7C,5D,39,8D,5,48,48,48
230 DATA 48,39,8C,2,DD,27,15,A6,
82,81,30,25,11,81,3A,25,C,81,46
240 DATA 22,9,81,41,25,5,8B,9,81
,4F,39,C6,8,D7,8C,7E,A9,51,7E,A9
250 DATA 28,7E,B9,9C,8D,F8,8D,26
,31,8D,FF,21,30,A8,D6,8D,F0,8D
260 DATA 1E,8D,12,C6,8,30,1,CE,1
,DA,A6,80,27,36,A7,C0,5A,26,F7
270 DATA 20,36,8D,D8,C,89,7E,A3,
90,86,A3,8C,86,C3,97,89,39,8D,C6
280 DATA 30,A8,ED,8D,F1,8D,C2,30
,88,DF,8D,BD,20,EB,8D,B9,1F,21
290 DATA 20,DB,8D,E8,96,44,BD,A2
,85,C,89,39,86,20,A7,C0,5A,26,FB
```

```
300 DATA 8D,D7,30,88,18,8D,C3,A6
,1,81,58,26,5,8E,2,20,20,7,17,FF
310 DATA 41,26,E9,9E,7C,BF,1,E5,
BF,2,1E,CE,2,2F,DF,45,C6,31,D7
320 DATA 44,33,8D,1,7F,DF,42,8D,
BF,30,A8,18,8D,B4,A6,1,81,58,26
330 DATA 39,DE,42,CC,0,19,ED,C4,
9E,45,ED,84,C6,20,ED,42,DC,19,ED
340 DATA 44,ED,2,DC,1B,ED,46,8D,
A,8D,8,C,44,8D,11,25,72,20,5E,DE
350 DATA 42,33,44,DF,42,DE,45,33
,42,DF,45,39,C,44,86,39,91,44,39
360 DATA 17,FE,DF,26,B5,DE,42,DC
,7C,ED,C4,17,FF,6D,30,A8,25,8D
370 DATA 79,30,13,17,FF,5D,17,FE
,C7,26,EE,DE,42,DC,7C,ED,42,17
380 DATA FF,55,30,A8,2A,8D,61,1F
,21,8D,5D,86,E3,97,89,BD,A3,90
390 DATA 17,FE,A9,26,E8,DE,45,DC
,7C,ED,C4,8D,B6,25,13,8D,A5,BD
400 DATA A9,28,30,A8,3C,8D,3D,BD
,A1,B1,81,59,10,26,FF,62,9E,45
410 DATA 6F,84,6F,1,FE,1,8F,FF,1
,FF,CE,1,E9,FF,1,8F,DC,74,7F,2
420 DATA 45,DE,42,83,0,E8,DD,7C,
30,1E,8C,2,2D,27,2E,33,5C,EC,42
430 DATA A3,C4,E3,84,10,93,7C,22
,17,20,EA,7E,B9,9C,8D,FB,17,FE
440 DATA CC,30,A8,74,8D,F3,BD,A1
,B1,81,D,26,F9,39,AE,84,30,1F,BF
450 DATA 2,46,7A,2,45,A,44,BD,A9
,28,17,FE,AA,BD,A7,CA,30,2A,8D
460 DATA D5,BD,A7,E9,BD,A9,28,17
,FE,9A,30,A8,5B,8D,C7,30,8C,6C
470 DATA 9F,42,8E,0,F,9F,7C,8E,1
,DA,9F,7E,BD,A7,E5,BD,A7,D8,8E,1
480 DATA 8E,9F,7E,8E,1,BA,9F,7C,
8D,2F,D6,44,C0,30,D7,44,8E,0,1
490 DATA 9F,7C,8D,22,9E,42,AE,84
,9F,7E,CE,1,FF,DF,7C,DE,42,EC,42
500 DATA C3,0,1,93,7E,27,F,10,83
,0,FF,24,2,D7,7D,8D,2,20,E2,7E
510 DATA A7,F4,0,7C,F,7D,8D,F7,9
E,42,30,4,9F,42,A,44,26,CD,BD,A7
520 DATA E9,BE,1,FF,BF,1,8F,16,F
D,FC
530 DATA 2,0,0,0,0,1,8E,35,10,B6
,2,45,27,E,FC,2,46,DD,27,DD,23
540 DATA 83,0,C8,DD,21,1F,4,8E,0
,0,BF,1,8F,CE,2,2F,AE,C4,9F,7E
550 DATA BD,A7,F,26,1C,D,7C,2A,F
,5,33,42,AE,C4,26,EF,BD,A7,E9,7E
560 DATA 0,0,BD,AD,21,9E,A6,30,4
,9F,A6,7E,AD,C0,7E,A6,19
```

gram whether in BASIC or assembly language. Rather than having a BASIC program that generates *Link*, it is more convenient to have *Link* recorded on tape directly as an assembly language program. Therefore, type the filename *Link* and ENTER it. This will initiate the process of using *Link* to record itself on tape.

The second prompt requests the entry address of the main program, which in this case is *Link*. All the required *Link* addresses have been provided in the REM statement of Line 10 of Listing 1. In accordance with that REM, type 1100 and press ENTER. You do not need to type &H in answering the prompt. The hexadecimal address 1100 is the address at which *Link* starts executing. If any of the characters of the ENTERed address are not a valid hexadecimal digit, a beep alarm will be sounded and the prompt will be repeated.

The next prompt asks for the first source address of the first record that you want on tape. Associated with each record are two sets of addresses — source addresses and destination addresses. The source addresses are the lowest (first) address and the highest (last) address of the record as it is presently located in RAM. The destination addresses are the corresponding RAM addresses into which you want the record to be loaded.

In the case of *Link*, only one record is involved; hence, you should type the first source address of *Link*. That is, type 1000 and press ENTER.

The third prompt is similarly answered by typing and entering the last source address 132B of *Link*.

You will probably want at least two versions of *Link*, one to be loaded in its present RAM location and one destined for high RAM. Therefore, for the former version answer the fourth prompt by typing 1000 and pressing ENTER.

Since you only need to specify one record for *Link*, answer the next prompt by pressing the 'Y' key to indicate yes.

Instead of recording *Link* immedi-ately following *GENLINK* on your cassette tape, it would be more convenient to record *Link* at the beginning of the reverse side of the tape. Therefore, flip the cassette over, rewind the tape and position it. Then answer the positioning prompt by pressing ENTER.

In accordance with the next prompt, depress the Play and Record buttons of your recorder and then press ENTER. When the recording is finished, the recorder will stop and the initial prompt of *Link* will return.

Now, you can repeat the process to produce a high RAM version of *Link*. Therefore, type the filename *HILINK* and press ENTER. The requested entry address of *HILINK* is the destination entry address which is 3DD4 or 7DD4 depending on whether you have a 16K or 32K RAM, respectively. The first and last source addresses that you must type are the same as previously, 1000 and 132B. The first destination address is 3CD4 or 7CD4 for a 16K or 32K system, respectively. When you later load *HILINK*, you will not have to use the *CLEAR* command to memory protect it. *HILINK* will be automatically memory protected.

A detailed example will be presented to illustrate how to use *Link*. However, before that presentation, it would be well to determine whether or not you have good recordings of *Link* and *HIL INK*. Do *not* use *SKIPF* to make that determination. Use of *SKIPF* on any *Link* produced recording will always yield an I/O Error message. *Link* purposely forces an I/O Error to occur as a means of altering the *CLOADM* command routine. *Link* changes the "hook" that links the ROM and RAM when errors occur. The new hook causes entry to be made to the first loaded record which controls the loading of all succeeding records. The original hook is restored before loading the subsequent records. Hence, those records are checked for I/O Errors as they are loaded.

To test the recordings of *Link* and *HILINK* do the following: turn off your computer and then turn it on again. Type *CLOADM* and press ENTER. Rewind the tape and position it. Finally, depress the Play button. While *Link* is loading, note that the letter 'F' at the top leftmost position of the screen stops blinking. The blinking of 'F' on all *Link* produced recordings will be suspended. The purpose of suspending the blinking of 'F' is to guarantee the unmarred loading of the text screen when you desire to precede the running of the main program with one or more screen messages or pictures. If the recording is good, no I/O Error message will occur. Furthermore, upon the completion of loading, *Link* or *HILINK* will automatically start and the initial prompt will appear on the screen. To exit from *Link* for the *CLOADMing* of *HILINK* press the Reset button. Before you load *HI LINK*, note the recorder counter setting for later reference.

If you should happen to have a bad recording of either *Link* or *HILINK*, *CLOAD* the program *GENLINK* and run it. Then type *EXEC*, press ENTER and repeat the process of recording *Link* and *HILINK* on a new tape.

The programs of listings 2 and 3 are integral parts of the example to illustrate how to use *Link*. Lines 10 through 70 of Listing 2 construct a text screen and transfer its contents to another area of RAM. Lines 80 through 100 produce a simple, high resolution graphics display. The remaining lines of Listing 2 generate a machine language routine and store it in RAM. *Link* will be employed in concatenating and recording the text screen, graphics screen, machine language routine and the BASIC program of Listing 3.

When you have typed the program of Listing 2 correctly, run it. You may wish to save it as a precautionary measure. After running the program of Listing 2, erase it via the *NEW* command. Then type Listing 3.

Line 10 of Listing 3 turns on the previously loaded graphics display. The remaining lines "paint" the display in a

```
Listing 2:
10 CLS:K=255
20 FORI=0TO31:POKEI+J+&H400,K:NE
XT
30 K=K-16:J=J+32:IFK>142THEN20
40 PRINT@238,"LINK";:PRINT@268,"
EXAMPLE";:K=K+32
50 FORI=0TO31:POKEI+J+&H420,K:NE
XT
60 K=K+16:J=J+32:IFK<256THEN50
70 FORI=0TO511:POKEI+&H2A00,PEEK
(I+&H400):NEXT
80 PMODE4:PCLS:SCREEN1,1
90 CIRCLE(128,96),85
100 PAINT(128,96),1
110 FORI=0TO23:READA$:A=VAL("&H"
+A$):POKEI+&H2D00,A:B=B+A:NEXT
120 IFB<>3116THENCLS:PRINT@267,"
DATA ERROR":STOP
130 DATA BD,B3,ED,DD,44,9E,BA,33
,89,18,0,DF,42,A6,84,98,45,A7,80
,9C,42,26,F6,39
```

variety of colors. The color changes are achieved primarily through the machine language routine called by the *USR* functions of lines 40 and 50. This routine is assumed by the program to have been loaded into the high RAM and automatically memory protected there. The example would be more realistic if the graphics screen had contained an intricate drawing requiring considerable program memory to produce it. In such a case the loading of the completed drawing would result in a significant savings in program memory. Frequently, the saved memory could be put to profitable use in program expansion and improvement.

Do not run the program of Listing 3 when you have finished typing it correctly. Instead refer to the previously noted recorder counter setting in positioning the tape for *CLOADMing HI-LINK. HILINK* rather than *Link* is used here because *Link* loads into the graphics screen memory area and would therefore ruin the display generated by the program of Listing 2.

The completion of the loading of *HILINK* is signalled by the appearance on the screen of the first prompt. Answer it by typing and entering the filename *EXAMPLE.* Usually the next prompt requires the typing of a hexadecimal address. There is one exception. That occurs when the main program is in BASIC, which is the present situation. In such a case, just press the 'X' key and then ENTER.

The text screen was stored by the program of Listing 2 in the RAM area from 2A00 through 2BFF. Hence, answer the next prompt by typing and entering 2A00. Similarly, type and ENTER 2BFF in response to the last source address prompt. Because you will want the text screen to reside in the usual location, answer the destination address prompt by typing 400 and pressing ENTER.

In order to specify the second record, press the 'N' key in response to the next prompt. The second record is the graphics screen. If you have a cassette-based system, the screen resides at addresses 600 through 1DFF. However, if you have the Disk BASIC ROM connected, the graphics screen is located at addresses E00 through 25FF. Thus, your response to the first source address

prompt should be the typing and entering of 600 or E00 depending on your system. Likewise, for the last source address prompt, type either 1DFF or 25FF and ENTER. In response to the destination address prompt type and ENTER 600 or E00 for cassette or disk-based systems, respectively.

". . . Link can be used as part of a scheme to protect your programs against piracy. There is a simple, yet fairly effective scheme for piracy protecting assembly or machine language programs."

Press 'N' to permit the specification of the third record. This record is the BASIC program of Listing 3. Typing and entering X will automatically take care of all address specification for you. Actually, an additional record will also be automatically specified. The additional record is only eight bytes long and consists of the vital BASIC program pointers at hexadecimal addresses 19 through 20 (corresponding to decimal addresses 25 through 32).

There is one more record to specify, so once again press 'N' in response to the record's specified prompt. Even though the previous record was numbered three, the present record has been given the number five. The number four record was the eight-byte record automatically specified along with the BASIC program. Record five is the machine language routine generated by the program of Listing 2. It was stored at RAM addresses 2D00 through 2D17. However, it is to be loaded into high RAM at addresses 3FE8 through 3FFF or at 7FE8 through 7FFF depending on whether you have a 16K or 32K RAM, respectively. Therefore, each of the next three prompts should be answered by

typing and entering, in order, one of the addresses: 2D00, 2D17 and 3FE8 or 7FE8.

Complete the process by pressing 'Y' and appropriately carrying out the instructions of the final two prompts. In positioning the tape make a note of the counter setting of the recorder for later loading of *EXAMPLE.* The signal that recording is finished is the return of the initial prompt to the screen. You will have a rather long wait for the prompt because of the 6K length of the graphics screen record.

In general, you may specify a maximum of nine records. If one of the specified records is a BASIC program, the most that you may specify is eight records unless the BASIC record is the ninth one specified.

Back to the example, load *EXAMPLE* by means of the *CLOADM* command. You should be quickly greeted with the text screen generated by the program of Listing 2. This screen will remain on display for the time needed to load the other records including the rather lengthy graphics screen. When loading is complete, the BASIC program will automatically start and the graphics screen will replace the text screen. The *USR* called machine language routine will keep changing the colors in the display. To end the program press the BREAK key.

For those with disk systems it is worthwhile interjecting a short note of caution. If you record a tape using *Link* with the disk ROM connected, always load the tape with a connected disk ROM. Likewise, if the tape is recorded with the disk ROM disconnected, always load it with the disk ROM disconnected; otherwise, problems would be likely to occur in the execution of the associated programs.

As was previously mentioned, *Link* can be used as part of a scheme to protect your programs against piracy. There is a simple, yet fairly effective scheme for piracy protecting assembly or machine language programs. The scheme will be illustrated by adding protection to *Link* itself.

With the present unprotected version of *Link,* the Reset button can be pressed to return to the CoCo's command mode in which an *EXEC* command can be employed to gain entry to a preloaded

program to analyze *Link*. In the proposed protected version of *Link* every BASIC command will be disabled and will result in an immediate error message when issued.

To add this protection to *Link*, turn your computer off and on again and load *Link*. Then give this version of *Link* the filename *PROLINK*. As you did previously, type and ENTER 1100 for the *Link* entry address. However, before specifying the *Link* program record, you must specify the protection record. It consists of six consecutive zero bytes. Locations 250 through 255 contain such bytes. Therefore, the first and last source addresses are 250 and 255, respectively, in the first record specification. For the first destination address, type 120 and ENTER it. Addresses 120 and 125 are usually stored the number of statements and functions, respectively, in the Color BASIC repertoire of commands. Making those quantities zero tricks the BASIC interpreter into "thinking" that it has an empty vocabulary.

Type 'N' to allow the specification of the *Link* program record. Carry out the remainder of the procedure exactly as you did in the production of the unprotected *Link*.

Some of you who are well versed in assembly language programming and are familiar with the CoCo's memory map may already see a way around this protection scheme. One of the hooks that link the ROM and RAM could be

when you are further protecting a program, 15E, 18D and 15E should be the first source, last source and destination addresses of the second specified record.

BASIC programs cannot be protected in the same manner as assembly or machine language programs. A BASIC program clearly could not run if its commands were disabled. The scheme

BASIC programs cannot be protected in the same manner as assembly or machine language programs. A BASIC program clearly could not run if its commands were disabled. The scheme for protecting any BASIC program does not disable the BASIC commands during program execution but does so when the program has been stopped by any means.

As in the more effective scheme for piracy protecting assembly language programs, the hooks at addresses 15E through 18D must comprise one protection record. However, one hook address in the record must be changed to point to a short machine language subroutine which forms a second protection record. The subroutine is what controls whether or not BASIC commands are disabled.

The objective of the program of Listing 4 is to generate the two protection records and store them in a convenient place in RAM. For purposes of illustrating the scheme, the area chosen to store the two records ws located at addresses 3000 through 303C. The first address 3000 was assigned in Line 10.

appropriately alters the hook record to provide entry to the short subroutine.

When you have correctly typed the program of Listing 4, save it for future use in protecting BASIC programs.

The protection scheme will be illustrated by applying it to the short example program of Listing 5. Therefore, after running the program of Listing 4, erase it by means of the *NEW* command. Then type Listing 5.

After typing the latter program, *CLOADM Link*. The responses to the *Link* prompts should be consecutively as follows:

```
PROBASIC
X
3000
302F
15E
N
3030
303C
3030
N
X
Y
```

Then appropriately follow the tape positioning and recording prompts. In the positioning process note the recorder counter setting for *PROBASIC*.

To test the protection scheme turn your computer off and then on again. Then load *PROBASIC* using *CLOADM*. When *PROBASIC* is loaded, it should request the typing of your

```
Listing 4:

10 CLS:A=&H3000
20 FORI=0TO47:POKEI+A,PEEK(I+&H1
5E):NEXT
30 FORI=0TO10:READD$:D=VAL("&H"+
D$):B=B+D:POKEI+A+48,D:NEXT
40 IFB<>977THENPRINT"DATA ERROR"
:STOP
50 FORI=0TO1:POKEI+A+59,PEEK(I+&
H168):NEXT
60 C=INT(A/256):POKEA+10,C:POKEA
+11,&H30+A-256*C
100 DATA 34,2,96,A6,81,6,25,FE,3
5,2,7E

Listing 5:

10 CLS:PRINT@226,"TYPE YOUR NAME
 & PRESS ENTER":PRINT@260,"";
20 LINEINPUTA$
30 PRINT@358,"PRESS ENTER TO STO
P"
40 K$=INKEY$:IFK$<>CHR$(13)THEN4
0ELSESTOP
```

the means of gaining entry to a preloaded program for analyzing *Link*. The occurrence of an error could be made to cause such an entry. Therefore, to make protection more effective you should specify a second protection record before the *Link* program record. This second record consists of the hooks located at RAM addresses 15E through 18D. There are other hooks but they have already been accounted for in the loader record which is always written on tape without your specifying it. Hence,

When you protect your own BASIC programs, you should (by appropriately editing Line 10) make the assignment consistent with the memory available to accommodate 61 consecutive RAM locations. Line 20 stores at addresses 3000 through 302F an image of the hooks at 15E through 18D. Lines 30 through 50 along with Line 100 are concerned with generating and storing the short subroutine. The subroutine is stored at addresses 3030 through 303C immediately following the hook record. Line 60

name. The program will then go into a loop. You can stop it by pressing ENTER, BREAK or Reset. Regardless of how you stop *PROBASIC*, typing and entering any BASIC command of your choice will cause the computer to hang up.

Link and the protection schemes were developed for your personal use. If you should wish to employ them commercially, please get in touch with me via THE RAINBOW to discuss mutually agreeable royalty terms.

An Introduction To The Inside Of The CoCo 2

Editor's Note: A correction to the wiring instructions and the BASIC listing for last month's column appear on Page 226 of this issue.

By Tony DiStefano
Rainbow Contributing Editor

My, doesn't time go fast? I can't believe I've been writing for RAINBOW for two years now.

January being THE RAINBOW's Beginners issue, I decided to introduce the novice to the inside world of the Color Computer. The latest CoCo 2 is the newest Color Computer to be introduced by Tandy. It is different inside from the old CoCo 2. You can tell the difference by the shield covering the power transformer. Though it functions the same, the insides of this CoCo are very different (again!). More on that later.

Before we get on our way, let me mention that I just came back from my second RAINBOWfest. I must say that these shows are great. I found THE RAINBOW staff to be very friendly and helpful. It is amazing to see that much enthusiasm generated about the Color Computer. Chances are I'll see some of you at the next RAINBOWfest, too, in California. Stop in and say hello. Look for me at the R.G.S. Micro booth.

(Tony DiStefano is well known as an early specialist in Color Computer hardware projects. He is one of the acknowledged experts on the "insides" of CoCo.)

Now, let's look into this little thing, but remember, opening your computer might void your warranty. Radio Shack only warranties the computer for three months, so after that you are on your own, anyway. First of all, *never* open the computer with the power on. Now that that's said, let us continue.

To open your CoCo, use the following procedure. Place the computer upside down on a towel (or other soft surface) on a clean work table. Remove the four screws (one in each corner) with a medium-sized Phillips screwdriver. There is one more screw to remove; it is behind the little sticker that says "Opening case will void warranty. See owner's manual for warranty informa-

tion." You must break this seal to remove the last screw. That is how Radio Shack can tell if you have opened it. Just push the screwdriver through the center of the sticker; it will give way to a hole. Some of the CoCo 2s may have a sixth screw on the other side. Remove the last screw. Turn the

Figure 1

TOO LITTLE   JUST ENOUGH   TOO MUCH

TABLE 1

IC #	Name	Description
1	SC77527	SALT Power supply and RS-232 CHIP
2	MC6821P	PIA Peripheral interface adapter
3	SC77526	DAC Digital to analog converter
4	NE555D	Timer for color burst in *PMODE* 4
5	74LS273	Octal D-Type Flip-flop
6	74LS244	Octal Buffer Driver
7	SC67331P	IIA Industrial interface adapter
8	MC6847P	VDG Video Display Generator
9	MC6809EP	CPU Central Processing Unit.
10	74LS02	Quad 2-input Nor Gate
11	74LS138	3 to 8 Decoder Chip
12	8040364B	ROM BASIC 1.2
13	8042364A	ROM EX BASIC 1.1
14-21	8040517	16K DRAM Dynamic Random Access Memory
22	MC6883P	SAM Synchronous Address Multiplexer

Beginners Project Parts List

Quantity	Description	Radio Shack #
1	LED	276-068 or 276-069 or 276-073
1	RESISTOR 1k ohms	271-8023

computer back right side up, and gather up the screws that drop out. Grab the top cover of the computer and pull it off. Wow! Look at all those things. The components marked with the letter 'U' (or 'IC' in the case of the newest CoCo) are known as ICs (Integrated Circuits). Table 1 labels all the ICs used in the computer and gives a short description of each.

Some of the components that make up the CoCo are very sensitive to static electricity. You must be careful not to zap (permanent damage caused by static discharge) a chip by touching the pins with your fingers. If you must touch a chip, always touch a ground point with your fingers first. This will discharge any static your body might be carrying to ground. A good ground point to touch is the RF adapter. That is the big metal can sitting to the left, where you plug in the TV wire. Another point is one of the metal clips that hold the bottom shield to the main PCB (Printed Circuit Board). You will find these clips all around the edge of the PCB.

Now that we have seen the insides of the CoCo and are a bit more familar with its parts, let's do something to it. About the simplest thing we can do is add a pilot light. It is not hard, and if you take it one step at a time, anyone will be able to do it, and the good thing about it is that it costs less than $1. By the way, this pilot light will work on any version, not just the CoCo 2. Before you plunge into this though, if you do not have any soldering expe-

rience, practice on something else first. To do this, you will need a soldering iron. A low power, medium or fine tip soldering iron will do. The solder to use must be a rosin core and not too thick. Radio Shack sells both at a reasonable price. If you have never handled a soldering iron before, get Radio Shack's proto-board and practice on it first.

Here are the step-by-step instructions on how to solder:

1) Make sure that your soldering tip is clean and hot. A wet sponge is great to clean the tip.

2) Secure the component to the PCB.

3) Heat the component and the PCB with the iron.

4) Touch the end of the solder to the component. My personal habit is to position the solder so that it will touch the iron, component and PCB at the same time.

5) When enough solder flows, remove the solder.

6) Remove the iron from the joint.

7) Wait until it cools before moving the component or the PCB.

To make a good joint takes practice. To put the right amount of solder also takes practice. Too much or too little could result in a bad connection. Examine Figure 1, and notice the difference between too little and too much solder. Sometimes a bad connection can be turned into a good connection just by heating up the joint again. After it cools, the joint should be shiny and smooth. Practice several times until you get the hang of it. There

is one more thing to remember; after all the soldering is done, clean the PCB with Radio Shack rosin cleaner-remover.

Now that you feel more at ease with soldering, it is time to put your newly acquired talent to work. Yes, the pilot light. There are only four parts to this project. The LED, a 1K (K=1000) ohm ½ watt resistor, and two short lengths of colored wire (preferably red and black). That is it. Examine the schematic in Figure 2. This is a diagram on how the components connect together and to the computer. The first thing to do is mount the LED. You must decide where to put it. After that, you must check that when mounted, it does not interfere with the normal operation of the computer, i.e., short out or lean on other components and does not prevent the cover from fitting properly.

Mount the LED by drilling a ¼-inch hole where the LED is to be mounted. Cut both sides of the resistor leads to about ¼ inch. Solder one side of the resistor to the long end of the LED. Solder one end of the red wire to the other end of the resistor. Solder one end of the black wire to the other (short)

end of the LED. Twist the two wires together lightly and cut them about 18 inches long. This should be long enough to have the cover out of your way if ever you want to open the computer again.

Figure 2

Now, solder the other end of the red wire to inside of pin 9 of the edge connector. That is the five volts side. How do you get to pin 9? Simple, just start counting from the end closest to the back of the computer. All the top pins are odd numbered, so count 1, 3, 5, 7, 9. Make sure that you don't short out two pins with the solder. Finally, solder the black wire to pin 33, count that one the same way. Pin 33 is the ground return pin.

Place the cover on top of the computer (without the screws for now) and turn the computer on. The LED should turn on. If not, chances are that you got the wires to the LED reversed. In that case, unsolder the resistor and the black wire to the LED and resolder them the other way. Otherwise, you should not have any problems. Tuck the wire in the cover and place the cover back on. Make sure that the wire does not stick out and that the keyboard is sitting on the pegs properly. Turn the computer over and replace the screws. There you are, your first modification to your computer. Now doesn't that make your day?

Getting The Most From Your Cassette Recorder

By Norman Latner

C hoosing a cassette recorder is, all too often, a hasty afterthought following the exciting and carefully thought out purchase of the Color Computer. However, after encountering endless I/O Errors and spending what seems like ages searching for programs located somewhere on your tape, you'll know you should have thought about it more.

Fortunately, the situation is not hopeless. You *can* make reliable, consistently high quality recordings by improving your present tape recorder and using good recording techniques and materials. And, if you're in the market for a new and better cassette recorder, I'll discuss how you can choose the most suitable unit.

Cassette recording with the Color Computer has a lot going for it. At 1500 Baud, it's five times faster than many of its competitors, allowing it to record an 8K program in less than 45 seconds. It's reliable and inexpensive. A brand name 60-minute tape can be bought for about a dollar in large metropolitan areas. It can store an amazing 675K bytes, or about 4 $\frac{1}{3}$ times more than a $5 Radio Shack disk. And while the cassette recorder can't immediately access a program located anywhere on the tape, or read as fast as the disk unit does, you will be able to locate programs fairly quickly using cue/review, speaker mute override and remote override features, which I'll discuss later. As we further

compare cassette to disk, we find that disks are far more prone to catastrophic failures. Don't ever turn the power switch off on a disk drive with a disk in place. Don't operate a drive in the presence of a strong RF field or any place where sparks from equipment turn-on might be close enough to disturb the unit. These disks may never again divulge their information to you. Disk drives can suddenly change from reading (playback) to writing (record) with no intervention on your part. The cassette unit, on the other hand, won't record unless you simultaneously press the Play and Record keys. And finally, a cassette recorder costs many times less than a disk system.

The Ideal Cassette Recorder

If you could get the ideal cassette recorder for use with the Color Computer, it wouild have the following features: cue/review keys, a remote override switch, a speaker mute override switch, a record monitor switch and a tape counter.

The value of the tape counter is fairly obvious. You can tabulate your programs by tape counter number and access them more quickly and accurately. By setting the counter to zero prior to a loading or saving operation, you can get back to the start of the program very easily if need be.

A record monitor switch, a feature available on a number of tape recorders, allows you to listen in while the recording is being made. Hearing the process can alert you to such

---

(Norm Latner, an electrical engineer, works in a government laboratory. He is involved in research and development of nuclear instruments, and is a frequent contributor to scientific journals.)

problems as recording with the computer in the high speed mode, (the "start tones" will sound much higher pitched than usual) or trying to record after erroneously typing *CLOAD* (there won't be any sound). It's reassuring hearing the two characteristic start tones followed by the crisp, static-like sound of the program code.

A speaker mute override, which could do you a great deal of good, is just another way of saying the cassette recorder's speaker is able to operate even though the plug in the ear-phone jack has turned it off. (That plug, of course, normally goes to the computer and allows loading.) The ability to hear the tapes loading is one of the most important aids available to you. It allows you to identify problem tapes by their muffled, mushy or wavering sound quality. It alerts you to programs accidentally recorded in the high speed mode. It lets you easily find the gap between programs. By locating and starting at this gap, you eliminate all those I/O Errors which occur when you attempt to load in the middle of a record. In addition, when used with the cue/review feature mentioned below, you can even hear the programs in fast forward or reverse. This allows you to move quickly through the tape while keeping count of the programs you've passed through. You can, for example, locate a program five ahead of your tape's current position by counting the noisy, followed by quiet, sequences. While it's theoretically possible to use *AUDIOON* to hear the cassette output through the TV speaker, there are some practical problems. First, an I/O Error resets this command, thus requiring constant re-entry. Second, the computer can't be doing anything else at the time — you can't set up to read another program while the first is still running. However, the speaker mute override is completely independent of the computer. The only reasona-bly priced recorder with this feature so far is Radio Shack's new CCR-82. If you're electrically knowledgeable, or can get help from a friend who is, it's easy to add this feature to your present recorder. It takes one resistor added to your recorder, or if you'd rather, you can build a separate box which gives you this feature plus a remote override switch. Details to follow.

A remote override switch will save you untold wear on your connecting cable, remote plug and jack as well as your patience. The remote jack, which permits the cassette unit to be turned on and off by the computer, also prevents you from operating the cassette recorder manually. By temporarily defeating this lockout, you regain manual control and can locate the tape anywhere you wish using Fast Forward, Reverse or Play. I've installed a switch to do this in my recorder or you can build it into the box mentioned above.

The final feature that you ought to have is cue/review. Cue/review keys, which are available on a number of cassette recorders, ordinarily operate like the standard fast

"You can make reliable, consistently high quality recordings by improving your present tape recorder and using good recording techniques and materials . . . Cassette recording with the Color Computer has a lot going for it."

forward/rewind. However, unlike these keys, cue/review also operates when the Play button is depressed, allowing you to hear the tape while fast forwarding or rewinding. This makes it possible to listen while the tape moves quickly, and then stop in the gap between programs, thus eliminating the bulk of I/O Errors.

And while we're talking about features of the ideal cassette recorder, we might add such niceties as a VU meter, or at the least, an LED, to indicate recording level. This helps you to arrive at the proper volume setting with a minimum of effort. A useful, but not essential feature is a linear slide volume control rather than the conventional rotary type. The setting of the linear control is obvious at a glance. End of tape-auto shut off is desirable, and by now, not too hard to get. However, be careful when using this, since the feature doesn't normally work in fast forward or rewind.

What You Can Buy . . . Or, The Realities Of The Marketplace

Now that we've talked about the ideal tape recorder, let's examine what's commercially available. The basic require-

ments for a CoCo compatible recorder are remote and earphone jacks, aux or line or mic jack and six-volt operation. (A recorder that takes four cells is a six-volt type.) These features are available and you should have little trouble finding a unit which also has a tape counter and a recording indicator. Cue/review keys are available on the somewhat higher priced machines, some of which also offer linear slide pots. The next step up offers units which usually include an AM/FM radio (which you don't need), along with monitor and ALC switches (which you do need). It might take a little looking, but you can find the close to ideal recorder in several name brands, at larger stores.

Radio Shack had not made any great effort to tailor their recorders, both the CCR-81 and the now discontinued CTR-80, to the needs of the computer user. For the most part, they resembled standard audio cassette units. A few of the features we'd like are there while others, which could have been easily implemented, are missing. Both recorders allow remote override during fast forward and rewind, but not during play, and both have a recording LED and a counter. The CCR-81 has the cue/review feature missing in the earlier model. The newest addition to Radio Shack's cassette recorder line, the CCR-82, has this as well as remote override and monitor switches. Although the Radio Shack recorder represents one of the easiest choices, you can do much better if you're willing to spend the time and effort. Aim for the unit which offers as many of the desirable features as possible and fits within your budget. As a final note, avoid like the plague, a cassette recorder which has single button recording rather than the standard type which must have both Play and Record keys depressed together. It's all too easy to slip up and destroy your favorite programs.

Improve Your Present Recorder

For those of you who want to improve your present recorder, the speaker mute and remote override functions can be added, and I'll describe how you can do it. While it's possible to add a recording monitor switch to most recorders, it's a lot more complicated and generally requires a schematic diagram, which puts it beyond the scope of this article.

There are two approaches to implementing the speaker mute and remote overrides, depending on your ambition and technical skill. In either case, you have to be able to solder, drill holes and otherwise work at a simple project level. If that's beyond you, or you can't get a technically inclined friend to help, then just skip to the next section.

The easier construction method consists of building a separate box which houses a small loud speaker, a switch, a resistor, two jacks and two plugs on short cables. A wiring diagram and suggested parts list are shown in Figure 1. There's nothing critical here, including parts layout. The switch can be any momentary type or, even better, a toggle switch with a center off, one side on, and the other side momentary. This arrangement frees you from keeping the switch held down during such chores as rewinding a tape. If you use this type of switch, connect the two outside lugs together and use this as one lead, with the center lug as the other. Another option you could consider is installing a foot operated switch, such as Radio Shack #44-610, at the end of a six-foot cable. This frees your hands entirely.
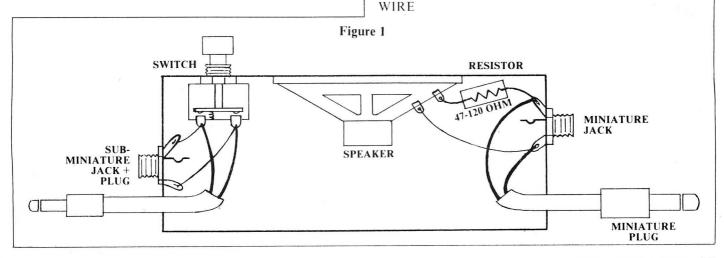
The particular value of the resistor (47-120 ohms) can be chosen to give a sound level that suits you. If yours is a quiet environment, you may want to use a value even higher than 120 ohms. To get the greatest flexibility, you could install a 500 ohm potentiometer in series with a 47 ohm resistor, allowing you to vary the volume as the need arises. While you can raise the resistance without problems, don't lower it much below 47 ohms, since this will decrease the signal available to the computer. Resistor tolerance, indicated by a gold or silver band, is unimportant in this application — use whatever you can get.

An important point to note is that you can't mount both jacks on a metal surface. The bushing of each jack is tied to one of its contacts, and this would short one jack to the other

PARTS LIST	
PROJECT CASE	RADIO SHACK #270-221 or 222 or 233
2″ SPEAKER	RS 40-245
1/8″ MIN PHONE PLUG	RS 274-286
3/32″ SUBMIN PHONE PLUG	RS 274-290
1/8″ MIN PHONE JACK	RS 274-251
3/32″ SUBMIN PHONE JACK	RS 274-292
RESISTOR	1/4 or 1/2 WATT 47 to 120 OHM (SEE TEXT)
WIRE	

Figure 1

Figure 2

— a definite no-no. Pay particular attention to the connections on the jack. Although there are generally three solder lugs, you only use two — the "sleeve" and "tip" lugs. The "sleeve" is the one that's electrically connected to the mounting bushing, and the "tip" contact can be identified by inspection. The third contact, not used here, allows for switching. See Figure 2.

A very convenient solution for building this box is to use an old transistor radio. Almost everyone has an old, non-functioning radio in his basement or the top of the closet. Remove the innards, leaving the loudspeaker and the earphone jack, which might be either a miniature or subminiature type. Install a second jack, of the other type, in a hole you either drill or melt into the case. Make two holes for the cables and another for the switch. Solder all the connections as per Figure 1. Your original computer cable, except for the microphone (or aux or line) input which stays put, now plugs into the box you've made. The cables and plugs from the box, in turn, go to the tape recorder, and you're finished. You now can hear the tape as it loads into the computer and move or play the tape without pulling out the remote plug.

If you have the skill, a neater and more compact way of adding these features is to modify the recorder itself. First, carefully take the back cover off the recorder and put the screws in a small container. (Be sure to get the ones that are invariably hidden in the battery compartment.) At this point, the earphone and remote jacks should be visible. Sometimes they are on a subassembly, which might require loosening a few more screws. Now examine the earphone jack and locate the "tip" and "switched" contact. See Figure 2. Solder the resistor to these points without removing any of the leads already attached. If the jacks are attached to the printed circuit board, trace the leads and solder the resistor to the appropriate pads on the board. Now locate the tip and sleeve contacts on the subminiature, remote jack. In this case, you connect two leads rather than the resistor. These leads go to the switch, which will be installed somewhere on the case, or you can use the foot switch mentioned earlier, instead. Almost any kind of small switch can be used and you can locate it anywhere you can find the room in your tape recorder. Around the battery compartment is usually a good spot, since battery operation with the computer is unnecessary and undesirable. Drill the mounting hole, install the switch, solder the leads and reassemble the case. Now you're all set to hear your programs playing through the recorder's speaker and to override the computer's control by operating your switch.

Making The Best Possible Recordings

To make a good tape recording, you need a good tape. Not only does the magnetic tape have to be uniform and of good quality, but just as important is the tape transport mechanism within the cassette. Poor corner guide rollers or none at all, ineffective anti-friction sheets, non-rigid cases, and poorly made center hubs that do not turn freely can cause tape jams, uneven spooling and tape binding. Magnetic tape problems usually result from non-uniform oxide coating and can cause dropouts and varying volume. All of this translates into I/O Errors and headaches. Fortunately, these problems can be avoided by buying name brand audio tapes or shorter computer tapes available from reputable dealers. The 60-minute audio tape is relatively inexpensive and a good size to use. The 90-minute tape can also be used, but avoid the 120-minute or longer sizes, since these tapes are thinner and more susceptible to stretching and breaking.

The cassette tape you end up with should have a case which doesn't deflect with moderate finger pressure; the window area near its hubs should be of rigid clear plastic, not thin film; the hubs should move freely when turned with a pen. If the case is transparent, you should see corner guide rollers, preferably set on metal pins. The pressure pad, which holds the tape against the head, should be made of felt rather than sponge rubber, which deteriorates with age. And finally, the case should be held together with screws rather than glue to facilitate repairs if needed.

If you're reusing a tape, it's best to bulk erase it before recording. Bulk erasing reduces the noise level below the recorder's own erase head level. If that's not possible, at least erase it as a separate step prior to use. Put it into your recorder or tape deck, lower the volume to minimum, the ALC to 'Off' if possible, and set the unit to 'Record.' This method assures a clean tape.

Don't neglect periodic maintenance of your cassette recorder. Clean the record and erase heads, as well as the pinch roller and capstan, using readily available head cleaner fluid and swabs. Solvent grade alcohol, which contains little or no water, can also be used, but avoid rubbing alcohol which is 30 percent water. If you can see any oxide deposits on your tape heads, your unit is overdue for a cleaning.

Demagnetizing the tape head is also a good idea and is recommended after every 10 to 15 hours of machine use. A magnetic bias on the head will cause an increase in noise level and a loss of high frequency response which can cause I/O Errors. Relatively inexpensive demagnitizers are available, including a very easy to use unit built into a cassette case. You simply insert it as you would any cassette, and then set the recorder on play for a few seconds.
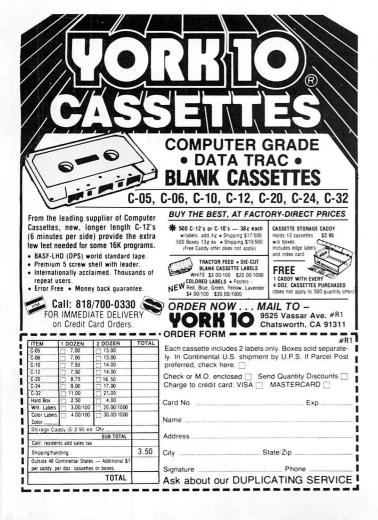
Automatic level control (ALC) is a mixed blessing. It relieves you of the burden of setting the recording volume, but it has some disadvantages. It takes about one-half second to operate, causing a sudden change in the volume of the starting tones at the beginning of each record. In addition, the setting it chooses may not be the optimum one. It's a compromise, but one that you can live with. However, if it's possible to switch the ALC off, do so for the best recordings.
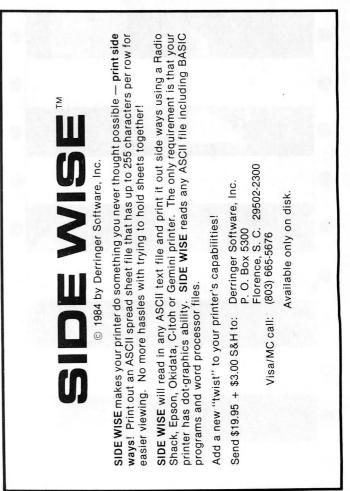
At this point, if your recorder doesn't have ALC, you'll need to determine the optimum volume control setting for recording. In general, the best results can be obtained by setting the level to record as high as possible with only occasional excursions into the overload region. This can be seen on the VU meter or recording LED if you have one. Without any indicator, you'll have to use a trial and error method. Start by setting the volume control to an arbitrary point, say three-quarters of the way up, and *CSAVE* a program. Next, remove the cassette and play the program back on a stereo tape deck (yours or a friend's). Watch the VU meters, or better yet, the dual series of LEDs, standard on newer decks. See if the recorded level is generally high, just below or occasionally going into the overload (red LED's) region. If so, you've been recording at the right level,

otherwise, make corrections and repeat the procedure until it's right. It doesn't take long. Once you've got the correct setting, mark the position with a dab of nail polish or paint, and always record at this setting.

While you're at it, you can also check your tape tracking. The cassette recorder used with your computer is monophonic and lays down a wide track which spans both the left and right channel tracks recorded by a stereo tape deck. Thus, when you play a monophonic tape on a stereo deck, the left and right VU or LED indicators should show an equal response if all is well. (Editor's Note: Many mono recorders normally record their single track off-center compared to the usual stereo track positions, so there can be as much as 3 dB of difference between the left and right channels when playing the tape on a stereo machine. In a few cases, you may be able to adjust head height to cure this problem, though it shouldn't have any ill effect when using the recorder with a computer.) If one side is noticeably different from the other, you have a problem. The cause could simply be due to oxide deposits on the recording head and easily cured by cleaning, or it could be due to misalignment of the recording head. This latter problem is sometimes spotted when a friend tries to read one of your tapes. Your tape, though it works well for you, gives him I/O Errors, and when heard on a good audio system has a mushy, wavering quality. You won't notice it because the error is present in both your recording and playback and thus tends to cancel out. However, this makes it difficult to share programs with friends. If you're technically inclined, you can readjust the alignment, otherwise get help (and skip the next paragraph).

The cassette head assembly generally has an azimuth

AND IF YOU REALLY WANT TO EXPAND YOUR COLOR COMPUTER...

If you've been plotting with your friends, trying to slip your dad a few hints about your Color Computer wish list, here's a suggestion:

Just ask for one program.

That's it. Just one. Just mention Zaxxon, Sega's exciting arcade-style game that puts you inside a daring space fighter. Or drop a few hints about Flip Side, an exciting educational game from Children's Computer Workshop—a division of the Sesame Street™ people.

Dad Will Take It From There
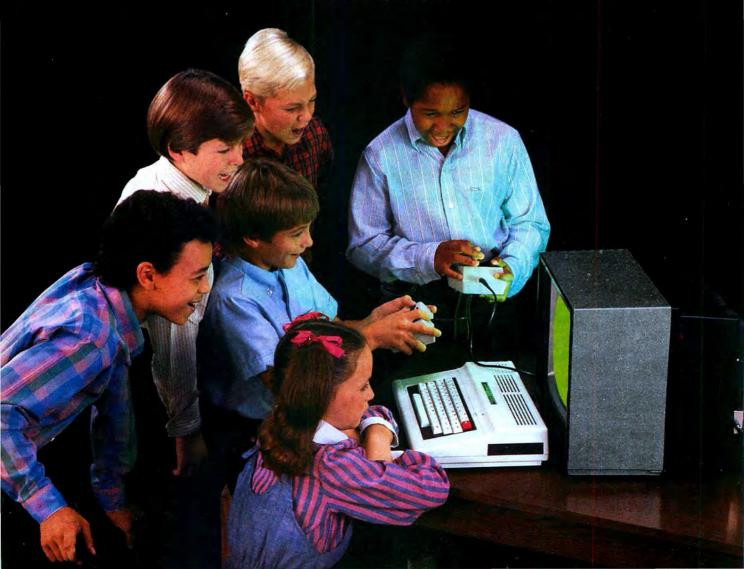
And don't worry, that's all it will take.

Once Dad tries to out-fly Zaxxon or out-smart Flip Side, he'll be hooked. And now's the perfect time for Dad to get interested, because your nearby Radio Shack is filled with software and accessories perfect for your whole family.

We have a complete selection of software from Spinnaker, a leader in thrilling educational software for children. And if you ever give her the chance, sis will love Facemaker (26-3166, $29.95), a fun way to learn about the computer while creating colorful graphics, Fraction Fever (26-3169, $29.95) and more.

From Walt Disney, we have exciting learning games with Mickey Mouse and Donald, including Mickey's Alpine Adventure (26-2534, $34.95), a fun way to learn about vowels, consonants and the rules of

...ASK DAD FOR JUST ONE PROGRAM. HE'LL GET CARRIED AWAY!

spelling, and Telling Time with Donald (26-2530, $34.95).

Of course, there are always the best in arcade-style games to choose from, including Androne (26-3096, $19.95), pitting you against Data Bugs inside your computer, and Color

Backgammon (26-3059, $17.95), the classic game of skill adapted for your Color Computer.

Software for Dad, Too!

Dad can choose Color SCRIPSIT® (26-3105, $34.95), a powerful home word processing program with advanced features, and Personal Finance II (26-3106, $34.95), perfect for home budgeting and financial graphing. And it's not all work and no play. Our new Trivia Fever (26-3295, $29.95) is fun for the whole family.

Add Accessories for Powerful Performance

Dad's sure to be tempted by our selection of convenient accessories, too, including the new TRS-80 Graphics Touch Pad (26-1185, $59.95). Use it with your

software to make color graphics at a touch, or use it with games as a fast-action game paddle. We have printers, disk drives, joysticks, cassette recorders— even our Color Mouse—to tempt Dad, too.

Drop a Hint Today

Don't wait! Let Dad know how easy it is to expand your Color Computer into a powerful home system and an educational learning tool. And tell Dad that Radio Shack honors six major credit cards, too!

Radio /hack®
The Technology Store™
A DIVISION OF TANDY CORPORATION

Prices apply at Radio Shack Computer Centers and at participating Radio Shack stores and dealers. Sesame Street is a registered trademark of Children's Television Workshop. Walt Disney and Walt Disney characters are registered trademarks of Walt Disney Productions. Spinnaker is a registered trademark of Spinnaker, Inc. ZAXXON is a registered trademark of Sega, Inc. Trivia Fever is a registered trademark of Professional Software, Inc.

Send Me a 1985 Computer Catalog!

Mail To: Radio Shack, Dept. 85-A-346, 300 One Tandy Center, Fort Worth, Texas 76102

NAME

ADDRESS

CITY

STATE

ZIP

PHONE

adjusting screw at one end, while the other end acts primarily as a pivot. The screw is usually accessible from the outside of the case, so you don't have to dismantle the unit. There is an access hole located between the operating keys and the cassette space, at the left side of the head mechanism. It is often covered by a plastic plug or label of some sort. The adjustment is made with the cassette unit playing a tape of known good quality, such as a selection recorded on a good tape deck known to be in alignment. Using a small Phillips screwdriver inserted through the access hole, adjust to get the cleanest, least distorted sound. When you think you've got it, make a recording and recheck it on the stereo tape deck for final verification.

Some cassette recorders have tone controls. If yours has one, set it to the highest (brightest) position when used with the computer.
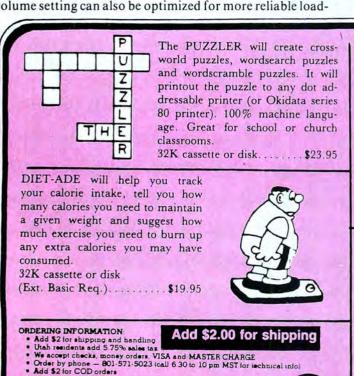
Although your own preferences should guide you, I find it best to leave a tape gap of four to five seconds between programs. With this length gap, you can usually rerecord a program in the same location after correcting the inevitable bugs. If the blank region is too short, it makes it difficult to locate the gap when searching through the tape. On the other hand, too long a gap makes it hard to tell whether it's really a gap or the end of the recorded area on your tape. This can lead to recording over, and thus destroying, one or more of your programs. To further safeguard your valuable programs, always punch out and remove the erasure prevention lug on the top left edge of the cassette after you've finished recording a side. You can always rerecord, if necessary, by putting tape over the opening.

Though not as critical as the recording level, the playback volume setting can also be optimized for more reliable load-

ing. Generally, there is a range over which you can get satisfactory results. Below this range, programs won't read in, while above, the computer will trip on noise and you may even hear sound breaking through the TV without *AUDIOON.* To define the range and then locate the best operating point, start at a low volume setting and try to read in a program. If this is too low, the steady 'S' at the top of the screen won't change. Raise the volume until you see the blinking 'F' when loading. This point, which you should temporarily mark with a spot of tape, is the minimum playback level. Now keep raising the volume until the computer responds to noise, either in the gap between programs or on a blank tape. At this level, the screen shows a reversed 'S.' Temporarily mark this maximum position. While the best place to operate within the range is the center, it's very convenient to use the recording setting if it's not too far off. Just leave the control at that point for both recording and playback. If you're not that lucky, paint another mark for the playback point and readjust between playback and recording when necessary.

Finally, a few hints on handling and storing tape are in order. Always avoid touching the tape with your fingers, since doing so might cause dropouts which can generate I/O Errors. When not in use, keep the cassette in its plastic case to protect it from dust and damage. Avoid prolonged exposure to temperatures above 110 degrees Fahrenheit, direct sunlight, high humidity and strong magnetic fields. Don't store them near heating appliances, TV sets or similar equipment.

Armed, as I hope you now are, go forth into the world of perfect recording.

16K
ECB

An Open-Ended Exploration

By Joseph Kolar
Rainbow Contributing Editor

The more you learn, the more you realize how little you know. That is true as far as the CoCo is concerned, and that is why every session at the keyboard is an adventure in learning.

The new CoCo owner bought his versatile machine for its graphics capabilities, among other things. We will explore the *POKE* and *PEEK* BASIC Statements as they apply to the text screen page.

I can't be sure what we'll do, but fire up CoCo and let us proceed line by line and investigate whatever comes to mind. It is a good, open-end way to learn and still have fun.

The text screen is what you see when you turn on the CoCo. It is your working area. *PRINT@* locations 0 to 511 cover all 512 locations on the text screen. Key in:

```
1 CLS
10 PRINT@ 10, CHR$(128)
100 GOTO 100
```

(Joseph Kolar is a free-lance writer and programmer dedicated to proselytizing for computers in general, and the CoCo specifically.)

Line 10 tells CoCo to print at the eleventh space of the top row, a black square. This is due to the first upper left-hand location being designated as 0. To verify that this is so, add and *RUN:*

```
11 PRINT@0,"12334567890";
```

Don't forget to add the semicolon. Now, delete the semicolon and see what happens. To help yourself learn, say to yourself, "Having deleted or omitted the semicolon, the black block, *CHR$(128)*, vanished. When the semicolon was restored, the black box was visible. So, what have I discovered?"

Insert an apostrophe or *REM* marker in front of 'P' in Line 11 and insert Line 9, copying the information in Line 11 ending with a semicolon. *RUN* and observe. Press BREAK, then delete the semicolon in Line 9 and *RUN.*

BREAK places the apostrophe (') marker in front of 'P' in Line 9. Delete the apostrophe in Line 11 and recheck both with and without the apostrophe.

You should have noticed that when Line 9 precedes Line 10, it is not necessary to add the semicolon. However, if you placed the information in Line 9 following Line 10, it is a different story! It is left to you to mull it over in your mind and figure out why this is so. There is no better way for a beginner to learn something than to work it out for himself.

When you are finished, you may *DEL9,* or if you prefer, keep it as a *REM* line in your program. It is harmless.

Please note that using *PRINT@* allows you to print a string of characters, such as 'RAINBOW' when enclosed in quote marks and separated from the location value by a comma. You can print the ASCII character codes using *CHR$(x).* The characters from 128 through 255 will create block graphics.

If you are unfamiliar with these graphics blocks, key in the following routine:

```
5 GOTO 200.
```

This line gets us around our routine, which we will refer to later.

```
200 FOR X=128 TO 255
210 PRINT@240, CHR$(X)
220 FOR Z =1 TO 200: NEXT
250 NEXT X
299 GOTO 299
```

Each *CHR$* character from 128 through 255 will be printed, in rotation, in the middle of the display screen. Add:

211 PRINT@270,X

This will give the numerical value of each shape displayed. It will also help you visualize each shape which will be directly above the second digit.

If you want to see the other characters, change Line 200:

200 FOR X= 33 to 127

CHR$(32) is a blank space, creating a space just as the space bar does. The low numbers are control codes and do not generate a visible display.

To keep this routine for later reference, put an apostrophe marker in Line 5. You will hold the routine harmless, but available.

Get in the habit of using the *REM* marker to hide or uncover program lines and routines. You will get lots of mileage out of this handy tool when you are experimenting or creating your own original work.

This *PRINT@* Text Screen can be accessed using *POKEs*. The memory locations of the Text graphics page begin at 1024. This memory location is equivalent to *PRINT@ 0*. The memory locations continue just as the *PRINT@* location and ends as memory location 1535, which is in the lower right-hand corner. It can also be called with *PRINT@ 511*.

Note that *PRINT@ 511-0* and memory location 1535-1024 both equal 511. Allow 1 for the location you are subtracting and you get 512, the total number of all possible locations.

Each one of these text screen locations may be accessed by means of *POKEx,y*, where 'x' is a specific location from 1024 to 1535 and 'y' is a value from 0 to 255.

So, what is *POKE* anyway? It is a statement that allows CoCo to place into a designated location whatever text screen character you desire. It has other uses not within the scope of this article. Press BREAK and add:

20 POKE 1066, 255 RUN.

This placed an orange block directly underneath the *PRINT@ 10*, black block.

PEEK allows you to look at a specified memory location to see what information, if any, resides there.

Press BREAK, *PRINT PEEK(129)*, and ENTER. This memory location is checked on I/O Error message when *CLOADing* a program from cassette. A zero means that memory is no good and if a one is returned, it signifies that the tape is no good. Try this:

PRINT PEEK(1066) ENTER.

The value of 96 is returned. This 96 represents 'blank' (empty). The reason for this is that we are not in the program, having broken out and location 1066 reverts to its original state, 'blank'.

The observant newcomer will notice that *CHR$(96)* is a reversed '@'. Verify this by unmasking Line 5. (Remove the apostrophe.) Mask Line 200 with '. Then add:

201 FOR X=96 TO 96 RUN.

This is a lazy person's way to substitute a single value in a *FOR TO* statement. If you used 201 X=96 you would still get the correct answer, but you'd also get an NF Error in 250. If this was an integral part of a real program, it would bomb out unless Line 250 was deleted.

This 'one value' hint is valuable when you may be experimenting with different values. OK! Press BREAK, mask lines 5 and 201 and unmask Line 200.

A disturbing fact remains. *CHR$(96)* and the 96 that was revealed by *PEEKing* at memory location 1066 are different. There are some differences between the ASCII characters using *CHR$* and the characters that CoCo recognizes from 0 to 255.

To compare the *POKEd* characters with the *CHR$* characters, change Line 200 and add Line 211:

200 FOR X=0 TO 127
211 POKE1269,X RUN

Let's make it neater. Press BREAK and change lines 210 and 211:

210 PRINT@234, CHR$(X)
211 PRINT@238, X

The graphics blocks from 128 to 255 are the same. If you want to check this out change Line 200 to include whatever values you care to compare.

Remember, the character displayed at the left, if any, is the ASCII code, and the character on the right is what CoCo will read for the same value when

it is *POKEd* into a memory location.

You are urged to make a reference table of the two sets of characters, side by side, insofar as they differ.

Now press BREAK, and mask Line 5 again.

To demonstrate that one set can be substituted over the other, Line 23 will *POKE* an orange box over the black box at *PRINT@ 10* and Line 24 will superimpose a black box over the *POKEd* orange box, using *PRINT@*. Add and *RUN*.

23 POKE 1034, 255
24 PRINT@42, CHR$(128)

POKEing graphics characters is one quick way to cover large areas. Press BREAK and add:

6 GOTO 300
300 FOR L=1024 TO 1055
310 POKE L,255
320 NEXT L
500 GOTO 500

This creates an orange line that covers all the text screen locations on the top row. Add:

330 FOR M=32 to 63
340 PRINTM, CHR$(175)
350 NEXT M

Using blue, *CHR$(175)*, we can use *PRINT@ M*, all locations in the second row to fill them in. *RUN* BREAK and to make a left border add:

360 FOR L=1024 to 1504 STEP 32
370 POKEL,255
380 NEXT L

Since we want only one vertical column, in Line 360, we start at 1024 and skip 31 columns to put a dab of color in each 32nd, or left-hand, row. Purists will note that we should begin with location 1056, but it is easier to go over the corner block in the top row.

Using *PRINT@*, we will create an orange border on the right-hand side. Press BREAK and add:

385 FOR M=479 TO 31 STEP-32
390 PRINT@M, CHR$(255);
395 NEXT M

We went from bottom to top for a change of pace. Note that we were unable to use *FOR M=511* etc. because filling in this corner box would cause the screen to scroll up one row. Omit

the semicolon at the end of Line 390 and watch a disaster area. We could fill that corner location safely with a *POKE* to avoid that pesky scroll. We do so when we create the bottom border. Press BREAK, add and *RUN*:

```
400 FOR L=1505 TO 1535
410 POKEL, 255
420 NEXT L
```

We can *PEEK(x)* a value while we are in the program. We will ask CoCo to check if memory location 1503 is orange, (255); *PRINT@ 237, "ORANGE"* and go ahead and create the bottom border. If 1503 is not orange, forget about the bottom border and skip to the end of the program. Press BREAK, add and *RUN*:

```
399 IF PEEK(1503)=255 THEN
PRINT237,"ORANGE"; ELSE 500.
```

To verify that this works, substitute 255 in Line 399 with another value and try it. As an alternate, pick an arbitrary *POKE* location, from 1024 to 1535 to see if it is orange.

You can *POKE* characters, other than the graphics values but, except for an asterisk or plus sign, which create neat borders or accents, it is silly to create a border of reversed @. Listing 1 will show an example of *POKEing* alphabetic characters.

At this time your mind is racing ahead with projects to try out. Before you do, put in the three missing blue sides of the inner border.

And, let's have some fun! Create a half-screen full of reversed @. Adjust the *POKEd* locations so the display is centered horizontally on the screen with a green band on the top and bottom. Open a partial row in the middle of the screen, leaving one blank space at each end and *POKE* your first name into the cleared space. Create a pause so your name may be read and then blank out the name slot with some graphics block.

Doing this exercise will give you ideas to either modify and improve what you have created, or go off into a frenzy of creativity in another direction.

Listing 1 is an example of using all *POKEs* to create a demonstration program which is somewhat similar to the exercise above.

One advantage of using the graphics characters, 128-255, is that you get to use all the colors available on your palette. You need not be an artist to have fun creating whatever your mind's eye conceives. You may wind up with some pretty impressive concoctions.

Some notes on Listing 1. Line 140 puts the top row of graphics characters on the screen the hard way — one at a time with an appropriate pause. Compare Line 140 with lines 180-210, which create the bottom segment.

There is no *RETURN* after Line 310, a *GOSUB* routine. This was a boo-boo. Since a similar routine follows, this effectively makes the pause 230 instead of 200. Can you see why? No harm was done and I failed to notice it.

Line 160 has no *GOSUB* pause between the two *POKEs* because they go onto the display as one unit. Line 230, the left border has a small pause between units so it blends nicely with lines 250-280, which override the text. Line 100 does not have the 'short' pause. It seemed to look better to have 0! come on as a single unit. The 'long' pause is used only before and after HELLO!

Read the listing and figure out what each program line does. Except for the two pause routines at the end, it is a

linear program and each routine follows exactly as it appears on the screen.

Hopefully, you will have some ideas to modify, expand or enhance this listing, so what are you waiting for?

Note Listing 2 should not be keyed in. Just compare it with Listing 1. It is the same as Listing 1 except it is tightened up using multiple program lines. Two changes, the missing *RE*

TURN was added to Line 310 and in Line 10, 20 was changed to 10 due to deletion of Line 20 from Listing 1 and subsequent UL Error message. Which listing would you rather key in?

Listing 1:

```
Ø  '<LISTING1>
10  CLS
20  C=RND(255)
30  IF C<144 THEN 20
40  GOSUB310
50  POKE 1260,96
60  POKE 1261,72:GOSUB320
70  POKE 1262,69:GOSUB320
80  POKE 1263,76:GOSUB320
90  POKE 1264,76:GOSUB320
100 POKE 1265,79
110 POKE1266,97
120 POKE1267,96
130 GOSUB310
140 POKE1226,C:GOSUB320:POKE1227
,C:GOSUB320:POKE1228,C:GOSUB320:
POKE1229,C:GOSUB320:POKE1230,C:G
OSUB320:POKE1231,C:GOSUB320:POKE
1232,C:GOSUB320:POKE1233,C:GOSUB
```

```
320:POKE1234,C:GOSUB320:POKE1235
,C:GOSUB320:POKE1236,C:GOSUB320:
POKE1237,C
150 GOSUB320
160 POKE 1268,C:POKE 1269,C
170 GOSUB320
180 FOR X=1301 TO 1290 STEP-1
190 POKEX,C
200 GOSUB320
210 NEXT X
220 GOSUB320
230 POKE 1258,C:GOSUB320:POKE125
9,C
240 GOSUB320
250 FOR X=1260 TO 1267
260 POKEX,C
270 GOSUB320
280 NEXTX
290 GOSUB320
300 GOTO 10
310 FOR Z=1 TO 200:NEXT
320 FOR Z=1TO 30:NEXT
330 RETURN
```

Listing 2:

```
Ø  '<LISTING2>
10 CLS:C=RND(255):IFC<144 THEN 1
0:GOSUB310
50 POKE 1260,96:POKE1261,72:GOSU
B320:POKE1262,69:GOSUB320:POKE12
63,76:GOSUB320:POKE1264,76:GOSUB
320:POKE1265,79:POKE1266,97:POKE
1267,96:GOSUB310
140 POKE1226,C:GOSUB320:POKE1227
,C:GOSUB320:POKE1228,C:GOSUB320:
POKE1229,C:GOSUB320:POKE1230,C:G
OSUB320:POKE1231,C:GOSUB320:POKE
1232,C:GOSUB320:POKE1233,C:GOSUB
320:POKE1234,C:GOSUB320:POKE1235
,C:GOSUB320:POKE1236,C:GOSUB320:
POKE1237,C:GOSUB320
160 POKE 1268,C:POKE 1269,C:GOSU
B320
180 FOR X=1301 TO 1290 STEP-1:PO
KEX,C:GOSUB320:NEXT:GOSUB320
230 POKE 1258,C:GOSUB320:POKE125
9,C:GOSUB320
250 FOR X=1260 TO 1267:POKEX,C:G
OSUB320:NEXT:GOSUB320:GOTO10
310 FOR Z=1 TO 200:NEXT:RETURN
320 FOR Z=1TO 30:NEXT:RETURN
```

32K
DISK

Block Out Those Troublesome Granules With DISKTEST.UTL

By Charles C. Zimmer

Did you ever pay the price of a good game program to buy a box of diskettes and carefully ration them out, only to find six months later one has a bad spot on it? You know if you take it back after six

(Chuck Zimmer is employed by Honeywell Information Systems, Inc. in Billerica, Mass. For the past 15 years he has worked in engineering, procurement and planning of all types of disk storage products.)

months you are going to get some raised eyebrows for claiming it had a bad spot on it when it was new. Or do you have some that have been used a lot on certain files to the point of producing errors, but the rest of the disk is practically new? You don't want to throw it away, but there's not a positive way to be sure you can avoid those bad spots.

DISKTEST.UTL can help you overcome these problems and feel better about those good disks you have. The

program gives your disks a thorough test by writing and reading all tracks with a variety of data patterns to check for errors. The test is contrived to produce the worst-case situations for data patterns and head positioning sequences so that it is not only a test of the diskette media, but is also a good test of the positioning accuracy of the drive.

If bad granules are found, the program writes a file on the disk called "DEFLOG.###" (### is the diskette

number) and all of the granules that contained an error are assigned to this file. This way they cannot be assigned to any of your files and therefore, are out of your way.

If you have a printer you are also· given a one-line report of the test results. Your printout sheet can be reinserted in your printer so you can maintain a continuous diary of the test history of your disk library. This can help you in the future if it is necessary to reinitialize a disk. If you do not have a printer, you are given the chance to write down the list of bad granules for future use.

Some Notes Of Caution

1) The program does a complete overwrite of the disk, including the directory track, so be sure you no longer need or have otherwise backed up all the files on the disk.

2) If you elect to use disks with errors "flagged-out" (meaning bad spots have been "marked" and made unavailable), please remember that the Radio Shack Disk Operating System command *BACKUP* will be aborted if a read error is encountered. This means you will need to back up individual files by loading them into the computer and saving them to another disk. You cannot count on being able to back up the entire disk at once with the *BACK UP* command. (See "Compatibility Note" for a way to overcome this limitation.)

How You Use Diskettes

Each individual needs to make his own decision about using disks with errors; this can be influenced by your overall procedures for handling files. In my case, I have a printer which produces the ultimate backup, but I also use cassettes for important files and do double program saves on my disks, plus periodical transferring of files to other disks.

I use flagged disks for program development where there is a lot of traffic, which tends to wear out the remaining good places. When a program is finished and put in service, it is transferred to a good working disk for regular use. If you have other utilities for recovering from disk crashes, this can make you even more comfortable about using flagged disks.

The balance of this article is written assuming you plan to save and use flagged disks so all of the procedures are discussed. If you choose to discard disks with errors, the program is somewhat simpler to run and you will be notified of all you need to know.

How The Program Runs

The program is self-prompting, and as it runs a series of messages will appear on Line 15, with prompts on Line 16 for the required responses. These appear whenever the program stops to accept an input from you.

An operating flow chart is shown in figures 1-a and 1-b. On the left are the messages and prompts which appear during the program. The information on the right side of the figures provides more explanation, and shows the program flow from stop to stop. The hi/lo legend beside the screen information tells which audio signal is given at the various stops. A high signal is to notify you that a normal stop has been reached. The low signal indicates an error condition. The program contains checks on inputs, so out-of-range inputs will not interrupt the program at critical places. Please note the disks must have been previously initialized with a *DSKINI* command.

The program begins by asking you to adjust the volume on the TV. Then you are asked to enter the drive number for the test, and to select between a short and a full test. If you want a printout of the results, you are next given the chance to input the day's date (six digit string). Printout or none, you then reach the main starting and stopping point called the DISK CHANGE point in Figure 1-a. Here, you insert the disk to be tested and when the test is completed, you will return to this point. You may then remove the disk and

Figure 1-a

```
                    PROGRAM FLOW CHART

     Screen Messages and Prompts        Program Stops
                                              v
| PLEASE TURN UP VOLUME ON TV |     VOLUME ADJ.  - Adjust volume for desired level
| (ANY) TO CONTINUE           | hi/lo v
                                              v
                                         on (ANY)
                                              v
| ? (number entered appears here)|   DRIVE SELECT
| ENTER DRIVE NUMBER (0-3)      | hi    v
                                       on (ENTER)
                                              v
|        (no message)           |   TEST SELECT
| (S)HORT TEST  (OTHER)FULL TEST | hi
                                       on (RESP)
                                              v
|        (no message)           |   PRINT QUESTION - Decide if printout is wanted
| (P) FOR PRINTOUT  (OTHER)CONTIN | hi   v-)-}-) on (OTHER) -)-)-)-)-)-)-)-)-)
                                              v                              v
                                         on (P)                             v
                                              v                             v
| ? (date entered appears here) |   DATE ENTRY  - To appear on printout  v  no printout
| ENTER DATE (YYMMDD)           | hi    v                                 v
                                       on (ENTER)                         v
                                            v(-(-(-(-(-(-(-(-(-(-(-(-(-(-(-(-(-
                                              v
| INSERT OR CHANGE DISKETTES    |   DISK CHANGE - We start and end here with disk change. (-(-
| (C)OMMENCE TEST     (Q)UIT    | hi    v                                                    ^
                                       v-)-)-) on (Q) -)-) to END and OK prompt.            ^
                                              v                                             ^
                                         on (C)                                             ^
                                              v                                             ^
| ? (number entered appears here)|   NUMBER ENTRY                                           ^
| ENTER DISKETTE NUMBER (###)   | hi    v                                                   ^
                                       on (ENTER)                                           ^
                                              v                                             ^
|    OPERATION: FILE CHECK      |       v  (Not a stop, Information only)                    ^
                                        v                                                   ^
                                        v                                                   ^
| DIR. TRACK ERROR-CAN'T CONTIN. |      v-)-) if read error -)-) DIR. ERROR  -) (ANY) -)-)-)^
| (ANY) TO RESTART OR QUIT      | lo    v                                                    ^
                                        v       (bad direcory track or unformatted disk)    ^
                                     no error                                               ^
./FILES PRESENT  LOG PRESENT  \.       v                                                    ^
'\NO FILES       NO DEFECT LOG /'  FILE CHECK POINT  - A chance to be sure you want          ^
| (C)ONTINUE   (OTHER)ABORT TEST | hi               to overwrite the entire disk.           ^
                                        v-)-)-) on (OTHER)  -)-)-)-)-)-)-)-)-)-)-)-)-)-)-)^
                                       on (C)
                                              v
                                              v
| ERROR IN LOG - WILL TEST ALL  |       v   (Not a stop - momentary error message if system
|       (no prompt)             | lo    v   cannot correctly decode an old defect log.)
                                        v
                                        v
                   TO FIRST DEFECT ENTRY POINT - see Fig. 1-b
```

insert the next for another test (repeating the test sequence) or quit, in which case the program ends in an orderly manner. When you continue from this point, the program makes a stop for you to enter the number of the diskette (up to 999).

After this, the program checks to see if files exist on the disk and also if there is a defect log file from a previous test. You are told of these findings at a stop called the FILE CHECK POINT. If you had previously tested the disk and it had defective granules, the program recovers this information by reading the old log and saving the numbers of the bad granules previously found. During the test sequence, the program will not take the time to retest the old bad granules because we want to be conservative and not think something that was once bad has become good again.

Perhaps the most important feature of the FILE CHECK POINT is when the program stops. The upper half of the screen is showing you the first sector of the directory entries (Sector 3 on track 17), and you will see your filenames to help you recognize that you have loaded a disk containing files; they will be completely overwritten if you proceed with the test. If you made a mistake, this is the time to abort the test (the prompt tells you how).

If everything is OK you press 'C' to continue. I have used 'C' where possible as the go-ahead response. This makes it easier to work your way through the prompts.

You then come to the FIRST DE-FECT ENTRY POINT shown at the top of Figure 1-b. Here you are shown a defective granule map. You can manually enter defective granule numbers that you want to avoid and they will be incorporated in the defect log. Time will not be spent testing them. You will enter the granules from an old printout or manual log if the disk has been so totally crashed that the old log can't be read, or if you have used a bulk eraser on the disk. Upon continuing from this point the actual testing finally begins.

During the test sequence the upper half on the screen is the actual disk I/O buffer, so you see the graphics representation of the patterns being written to and read from the disk. The lower half of the screen contains a table giving the progress of the test, and the area of the disk being tested at the moment.

You should know two things about the test cycle. First, the program uses the Radio Shack *DSKCON* ROM routine to actually perform the test.

This is a powerful routine which not only performs seeks and reads or writes, but also performs retries in the event of an error. Many times you can tell when a read error occurs because the drive "recalibrates" (positions to track 0 and then back to the commanded track) four times in an attempt to perform the operation successfully. If after four attempts it is unsuccessful, an error code is created indicating the granule is bad. "Transient" errors, where the operation is sucessfully performed within the four tries, are not counted as "hard errors" and therefore are not logged.

If you notice the granule number during transient errors, you may wish to manually flag this granule bad at a later point in the program.
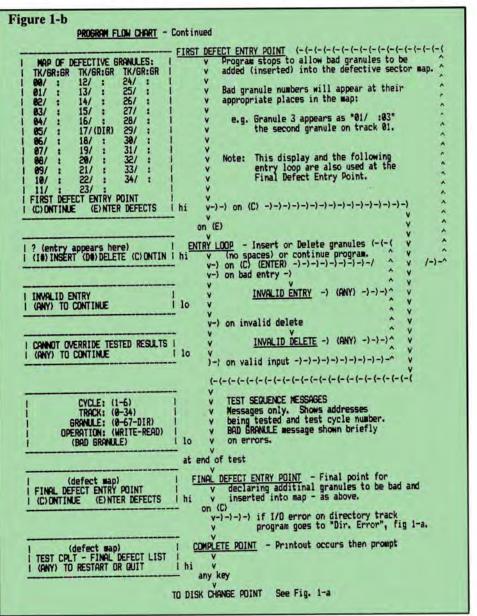
The second thing about the test cycles is that watching the screen and listening to the rhythmic operation of the drive can be hypnotic, so if you fall asleep please don't send me a bill for lost computer time.

The testing takes from four to 10 minutes, depending upon an option discussed later.

After the test is finished, you come to the FINAL DEFECT ENTRY point and are shown the defective granule map again (new bad granules are in reverse video). You may then manually flag additional granules as bad. This is another important step in the program for the following reason: if there is a bad spot on the surface of the disk, how do you know how big it is and how do you know that it lies only in the center of the track where the error occurred? The answer is you don't know how big it is, nor do you know that the edge of it doesn't extend a little over onto a neighboring track. The adjoining track works at this time, but may cause an error later.

To handle this little trauma and get

Figure 1-b

PROGRAM FLOW CHART - Continued

```
| MAP OF DEFECTIVE GRANULES:        |      FIRST DEFECT ENTRY POINT  (-(-(-(-(-(-(-(-(-(-(-(-(-(-(-(
| TK/GR:GR  TK/GR:GR  TK/GR:GR      |   v  Program stops to allow bad granules to be        ^
| 00/  :    12/  :    24/  :        |   v  added (inserted) into the defective sector map.  ^
| 01/  :    13/  :    25/  :        |   v                                                    ^
| 02/  :    14/  :    26/  :        |   v  Bad granule numbers will appear at their          ^
| 03/  :    15/  :    27/  :        |   v  appropriate places in the map:                    ^
| 04/  :    16/  :    28/  :        |   v                                                    ^
| 05/  :    17/(DIR) 29/  :         |   v  e.g. Granule 3 appears as "01/  :03"              ^
| 06/  :    18/  :    30/  :        |   v       the second granule on track 01.              ^
| 07/  :    19/  :    31/  :        |   v                                                    ^
| 08/  :    20/  :    32/  :        |   v  Note:  This display and the following             ^
| 09/  :    21/  :    33/  :        |   v         entry loop are also used at the            ^
| 10/  :    22/  :    34/  :        |   v         Final Defect Entry Point.                  ^
| 11/  :    23/  :                  |   v                                                    ^
| FIRST DEFECT ENTRY POINT          |   v                                                    ^
| (C)ONTINUE   (E)NTER DEFECTS  | hi    v-)-) on (C) -)-)-)-)-)-)-)-)-)-)-)-)-)-)-)           ^
                                        on (E)                                               ^
                                          v                                                  ^
| ? (entry appears here)            |   ENTRY LOOP - Insert or Delete granules (-(-(         v  ^
| (I#)INSERT (D#)DELETE (C)ONTIN | hi  v   (no spaces) or continue program.      ^  v        ^
                                       v-) on (C) (ENTER) -)-)-)-)-)-)-)-)-/     ^  v  /-)-^
                                       v-) on bad entry -)                       ^  v
| INVALID ENTRY                     |   v        INVALID ENTRY -) (ANY) -)-)-)^  v
| (ANY) TO CONTINUE             | lo   v                                       ^  v
                                       v-) on invalid delete                   ^  v
| CANNOT OVERRIDE TESTED RESULTS |     v        INVALID DELETE -) (ANY) -)-)-)^  v
| (ANY) TO CONTINUE             | lo   v                                          v
                                       )-) on valid input -)-)-)-)-)-)-)-)-)-)-^  v
                                       (-(-(-(-(-(-(-(-(-(-(-(-(-(-(-(-(-(-(        v
                                          v
| CYCLE: (1-6)                      |   v  TEST SEQUENCE MESSAGES
| TRACK: (0-34)                     |   v  Messages only.  Shows addresses
| GRANULE: (0-67-DIR)               |   v  being tested and test cycle number.
| OPERATION: (WRITE-READ)           |   v  BAD GRANULE message shown briefly
| (BAD GRANULE)                 | lo   v  on errors.
                                        at end of test
                                          v
| (defect map)                      |   FINAL DEFECT ENTRY POINT - Final point for
| FINAL DEFECT ENTRY POINT          |   v  declaring additional granules to be bad and
| (C)ONTINUE   (E)NTER DEFECTS  | hi    v  inserted into map - as above.
                                        on (C)
                                          v-)-)-)-) if I/O error on directory track
                                          v        program goes to "Dir. Error", fig 1-a.
| (defect map)                      |   COMPLETE POINT - Printout occurs then prompt
| TEST CPLT - FINAL DEFECT LIST     |   v
| (ANY) TO RESTART OR QUIT      | hi    any key
                                          v
                                        TO DISK CHANGE POINT   See Fig. 1-a
```

some sleep at night, I arbitrarily decide the two granules physically adjacent to the one having the error are also bad, and I manually flag them as such at this point. The granules you will flag are numerically two granules above and below the bad one so that, for example, if granule 50 is bad you will enter I48RETURN and I52RETURN when in the ENTRY LOOP of Figure 1-b. If you are not having a printout, this is the time to manually write down the defective granule numbers in a media log of some sort.

Upon continuing from the final defect entry point, you will see the file allocation table being created (if there are bad granules). You will then see a directory entry created for the DE-FLOG.###" file; it will be "killed" (first character set to zero) if there are no bad granules. There will be a pause and printout (if this is elected), and the program returns to the DISK CHANGE POINT from where you end or begin another test.

Initialization Notes

Page 58 of the Radio Shack disk manual describes bytes 0-55 of the sector as "system controls" information. Technically, this is a "header" field which is written only when the disk is initialized, and it identifies the upcoming sector number to the controller. The controller always reads this header before reading or updating the data field. In running *DISKTEST.UTL,* you may detect errors not caused by defects in the media, but because a system glitch

---

Figure 2

```
                 DEFECT LOG PRINTOUT SUBROUTINES (OKIDATA 92 Codes)

line 225 ---------------- PRINT ID SUBROUTINE  - BEGINS WITH A PRINTING DRIVER

                         Set computer baud to 2400 baud or for these rates:  300  600 1200
POKE 149,0:              Set MSB of baud rate in computer IPOKE 149,)          1    0    0
POKE 150,18:             Set LSB of baud rate in computer IPOKE 150,)         202  180   87
POKE 155,96:             Sets line width = 96 in CoCo      IPOKE 155,80 for 80 col-only
PRINT#-2,CHR$(28);       Sets printer for 12 cpi           IDelete for 80 col-only
         CHR$(27);CHR$(48); Code to set draft printing mode IDelete for single mode printers

line 230 ---------------- PRINT IDENTIFICATION INFORMATION

PRINT#-2,USING"X %d#XX X     X ";RIGHT$(LN$,3),D1,T$,D$;:
             I IIIIIIII    I^--Print space for legibility
             I IIIFIII^^^^^^---Print date (6 digits)
             I IIIIII^---------Print space for legibility
             I III^^-----------Print Sh or Lg for test type
             I I^^-------------Print d followed by drive #
             ^^^--------------Print diskette number

         Note:  If 80 col-only there will not be enough room on the line for all of
                these items.  Substitute PRINT#-2,USING"X X X     X ";RIGHT$(LN$,3),D$;:

RETURN                   Return from PRINT ID subroutine

line 235 ---------------- PRINT LOG SUBROUTINE  - BEGINS BY
                          PRINTING A CHARACTER INDICATING THE CONDITION OF THE GRANULE

X2=0:                    Clear gran counter to zero
FOR X=0 TO 67:           Start of loop to scan granule array

     IF G(X)=&HB9 THEN PRINT#-2,"-";     (-) for a granule found bad in a previous test
ELSE IF G(X)=&H99 THEN PRINT#-2,CH$(34); (") for grans you declare bad in in this test
ELSE IF G(X)=&HE9 THEN PRINT#-2,".";     (.) where an error occurred in this test
ELSE PRINT#-2,USING"#";X2;               (least sig. digit of gran #) for good granules

line 240 ---------------- PRINT MOST SIGNIFICANT DIGIT OF GRANULE # After each 10 granules

         Note:  If 80 col-only there is not enough room on the line for these extra
                tens digits.  Delete all PRINT#-2 statements this line.

X2=X2+1: IF X2=10 THEN X2=0:  Increment gran counter. If 10 done reset the counter and
                              print a superscripted, bold, double width "Tens" digit.

PRINT#-2,
    CHR$(27);CHR$(74);       Set superscript printing
    CHR$(28);CHR$(31);       Set double width print mode
    CHR$(27);CHR$(84);       Set emphasized (bold) print
PRINT#-2,USING"#";X/10;      Print the "Tens" digit for the next ten granule numbers
PRINT#-2,                    Start of control codes to restore normal printing
    CHR$(27);CHR$(73);       Stop emphasized printing
    CHR$(28);                Set 12 cpi printing again
    CHR$(27);CHR$(75);:      Stop superscript printing
NEXT ELSE NEXT               Go to next granule in FOR loop

line 245 ---------------- COMPLETE PRINTING OF DEFECT LOG

PRINT#-2,"": RETURN          Since all print commands used (;) for no carriage return &
                            line feed you provide a CR/LF here, then end subroutine
-------------------------------------------------------------------------------------
Sample of log printout with 12 cpi matrix printer:

101d0Lg 831124 0123456789¹ 0123-56789²0123".7"9³01234567894⁴01234567895⁵01234567896⁶01234567

-------------------------------------------------------------------------------------
Sample of log printout with 10 cpi 80 col-only printer:

101 831124 0123456789012 3-567890123".7"901234567890123456789012345678901234567
-------------------------------------------------------------------------------------
```

— or yes, operator error — causes header areas to be garbled (written over). If you suspect this may have happened, you can reinitialize the disk *(DSKINI)* to rewrite these areas anew. This is one case where you may find that disks once bad do become good again. Bulk erasing before *DSKINI* will really give a clean start.

Compatibility Note

DISKTEST.UTL is compatible with the *Disk Backup Utility* by Roger Schrag in the December 1983 RAINBOW, in that his program will back up disks which have been tested with *DISKTEST. UTL*. The good sectors in the granules flagged by *DISKTEST.UTL* will be copied to the backup disk, and the sectors having errors will be bypassed by the backup program. After running the backup, you may want to *KILL* the DEFLOG.### file which was transferred to the backup disk and is irrelevent on the backup copy. As mentioned in the "Backup Utility" article, you cannot back up to a flawed disk, so you should use only disks found error-free by *DISKTEST.UTL*.

Program Preparation And Options

There are a few things to go over before you begin typing in the program (Listing 1). Some of the points are technical, relating to your system configuration, and some are "cosmetic" options allowing you to adapt the program to your preferences.

First the functional subjects:

Memory Requirements

If you have a 32K machine the complete program will fit easily, including comments so you can go on to the next point. If you have a 16K machine all of the functional program will fit, but it will be necessary to delete some non-essential information. You should delete all lines beginning with ' starting at Line 30, delete all but the program name in Line 10, and delete the *PRINT* statement in Line 320. There is another option for 16K users to gain additional memory, and that is to insert a *PCLEAR 1* statement at the end of Line 15. This will free up an additional 4608 bytes (three graphics pages), which is more than adequate for the complete program, including comments.

A *PCLEAR 1* can make it necessary to cycle the power switch off and on in older machines in order to regain all four graphics pages for running your next program. Do not clear all graphics pages because special machine language routines are stored in graphics Page 1.

Systems With No Printer

If you wish, you can enter all of the code, including the printer sections, in anticipation of the future addition of a printer. You can avoid unnecessary menu steps dealing with the non-existent printer by changing the end of Line 315 from . . . . *GOSUB265:GO SUB200:GOTO350* to the following: . . . . *GOSUB265:P=0:GOTO350*. Technically, all that is necessary is to delete the *GOSUB 200* since all BASIC variables are initialized to zero at *RUN*, but this just seems a little more positive to me. If you do not wish to bother with the printer code at all, it may be omitted as follows:

Omit lines 195-205
Omit lines 220-245
Omit the GOSUB 200 in Line 315
Omit the IFP=1. . . .ELSE code in 330
Omit Line 620

Adapting To Your Printer

The program as listed contains printer driver codes and control sequences required by my printer, an Okidata Microline 92, and uses 12 cpi (96 column) printing. To assist you in adapting to other printers, Figure 2 gives a statement-by-statement description of the defect log printout subroutine. This explains what each step does so you can follow the same logical sequence using control codes for your printer, and come up with the same results. Special instructions are included to cover printers that are 80-column only. A sample of the output is shown at the bottom of the figure.

Creating Your Own Report

If you want to have an output that is less cryptic than my one-liner, you can write your own report subroutine to replace lines 225-245. Subroutine 225 is called at the end of the program in Line 620 and works with the following variables.
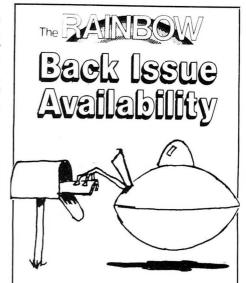
ARRAY G(69) in which the first 68 elements 0-67 contain one of four values representing the condition of the granules of the disk: Hex FF (OK), Hex B9 (bad from a previous test), Hex 99 (tested OK but declared bad by you in this test), and Hex E9 (bad due to an error detected during this test).

DN variable containing diskette number.

D$ string variable containing the date.

D1 contains the device number for the test.

T$ contains the test type,"Sh" or "Lg" for short or long test.

Length And Comprehensiveness

The *DATA* statements in lines 460 and 465 control the data patterns and the sequence in which the tracks are tested. A longer test gives more confidence in the results than a short test, and the program provides you the option of making a choice between time and comprehensiveness. Line 460 does a rotating, worst-case pattern test (about six minutes), and Line 465 does a 1,0 test (about four minutes). Either alone is a complete test and both together are quite comprehensive. There is more information on these tests in the technical details section for those so inclined. If you think 10 minutes per diskette is a good investment, then both 460 and 465 should be used.

Drive Number

This provides for entering the number of the drive to be used for the test. If you have a single-drive system and wish to avoid this unnecessary menu step, you can delete the *GOSUB 255* in Line 315. You can also delete lines 250 and 255 entirely, but this is not necessary.

Now a couple of fun options:

Background Color

The screens are formatted with information fields blocked out in green according to the way CoCo prints to the screen. You can change the surrounding background to any of the eight-plus black colors by substituting your color code in the Line 20 statement BC=3. For a black background color, we need to fool CoCo and use a value for BC = 1/16. Color code 3 (blue) is in the program because it produces a nice gray-scale contrast on a black and white TV. Happy viewing.

Audible Signals

Two strings are played at appropriate times, E$ for error conditions and A$ for attention points (usually stops). These reside in Line 285 and sound like crickets. If you would prefer something different, please feel free to wax musical and redefine these strings to suit your tastes. Just keep in mind that we're close to the memory limit in 16K, and also that a long string may make it necessary to adjust the amount of string space *CLEARed* in 15.

Debug Aids

Here are some suggestions to help you debug the program.

Scratch Disks

The first and most important thing to remember is that after you have typed the program, save it to disk and then take the disk out and insert a scratch disk to use for debug. Do *not* make your test runs with the disk containing the program in the drive because the execution of the program overwrites the disk! Another reason to use a scratch disk for debug is that the program contains machine language routines which are poked into memory. Errors in the ML code, or errors in the poking code, can cause CoCo to crash. The voice of experience speaks.

Shortened Test

The second tip is to shorten the test cycle so you can get all of the menu and program control code working without long waits for the test process. Listing 1 contains a "commented" Line 475 which does a short read of tracks 0-3. This is only for use during debug, but will not hurt anything if executed during actual running of the program. Line 490 causes the program to skip over the test codes contained in Line 460, so for debug, Line 490 can be changed to make it skip even more codes. For a short debug run remove the ' in Line 475, change the *TO70* in Line 490 to *TO120*, and use the short test option. This should reduce the "test cycle" to less than 10 seconds. When finished debugging, don't forget to re-comment Line 475 (insert ' before *DATA*), and change the Line 490 *TO* value back to 70.

Forced Conditions

There are a number of branches in the program that may never be used unless you encounter the right conditions. Many of these deal with error processing. You can force these branches by temporarily editing the program to insert the right conditions at the branch points. You should observe the rule to do only one at a time and be sure to delete the temporary code before going on to the next branch. Here are the places to modify:
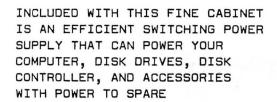
Insert		
EC=1:	after GOSUB35:	in Line 365
Insert EC=1:	after GOSUB35:	in Line 390
Insert EC=1:	after EC=USR2(X3):	in Line 535
Insert EC=2:	after EC=USR2(X3):	in Line 535
Insert EC=1:	after NEXTX4:	in Line 590
Insert EC=1:	after NEXTX4:	in Line 605
Insert X2=100:	after G(X1)=&HB9:	in Line 430

Technical Information

The information that follows is provided for persons wanting to know more about the program and the testing of magnetic recording media. This information need not be read to enter and use the program.

Recording Codes

When the drive records data on the disk, what it does is magnetize the iron oxide material moving under the head in a direction along the track. As the stream of bits is fed to the drive, the direction of the magnetization is switched back and forth along the track according to an encoding algorithm. The effect is equivalent to producing a series of small bar magnets laid end to end along the track, except they vary in length and each successive one is turned around (of opposite magnetic polarity) with respect to its neighbors.

Later on, when reading the track, these small magnets are moved under the head and each time the junction between two adjoining magnets passes under the head the magnetic flux through the head reverses, causing a small voltage to be induced in the head. This signal is amplified and processed to become the data sent back to the computer. What is important is the time sequence of these "flux reversals," because this is what determines if they are translated into ones or zeros.

The point of this discussion is if we want to test the media, what we want to do is find data patterns that let us

put flux reversals at all of the places they can be, and then read the disk to see if they are properly recorded. Some patterns of flux reversals are harder to recover than others, so our test can include data that will produce these worst-case patterns as well.

The main reason some patterns are hard to recover is because if the magnets are not all the same length, the short ones, being crowded and under more pressure, tend to get longer making their long neighbors shorter and shifting the point at which the flux reversal occurs. This produces what is called "peak shift" in the read signal which causes timing and data errors.

The standard data pattern used to create a worst-case peak shift is a repetitive Hex DB6. Since this is a 12-bit value it is sent to the drive in a 3-byte sequence, DB 6D B6, etc. Figure 3 shows five data patterns used in this test, the first of which is the DB6 pattern. For each pattern there is a line of asterisks representing where the flux reversals would occur along the track when the pattern is recorded. The symbols '(' and ')' show the direction of peak shift due to the magnetic crowding described above.

If you look over the set of patterns, you will see places where flux reversals can occur that DB6 does not test. It is for this reason that multiple patterns are used. Patterns 6DB and B6D together with DB6 produce what is known as a rotating worst-case pattern test, wherein all data 1 bits are subjected to peak shift in both directions.

In contrast, the Radio Shack disk initialization routine writes a repetitive FF pattern which tests all data 1 bit positions in one pass, but does not introduce peak shift. For a one-pass test this is a good choice, but more passes provide better comprehensiveness.

The 00 pattern also has uniform spacing (no peak shift), but you will notice that it is not in phase with the locations produced by an FF pattern. The 00 pattern essentially tests another whole set of flux reversal locations.

By using all five patterns, *DISKTEST .UTL* provides a quite comprehensive test of the media.

Other Causes of Errors

Data error problems can result from things other than imperfect media; here are a few other causes. The first is "adjacent track interference," meaning that the head is picking up some signal from the next track which is hindering its attempt to read the track it is on. This can be caused by wear, friction or looseness in the head positioning mechanism which prevents the head from stopping exactly in the center of the track.

An inadequate overwrite capability can also be a problem. When data is written, the head actually records the new data onto the track by forceably remagnetizing the oxide surface. If the write current is not correct, the head may not be strong enough to completely remagnetize the track, meaning that remnants of the old data could be

---

Figure 3

```
        TEST PATTERN CODING AND RECORDING

Pattern  0 is a repetitive Hex DB6 sent to disk as a repeating 3 byte sequence = DB 6D B6 etc.
Pattern  1  " "     "      " 6DB  "   "   "   " " "     "     "      "           = 6D B6 DB etc.
Pattern  2  " "     "      " B6D  "   "   "   " " "     "     "      "           = B6 DB 6D etc.
Pattern  5  " "     "      " 00   "   "   "   " " "     "     "      "           = 00 00 00 etc.
Pattern  8  " "     "      " FF   "   "   "   " " "     "     "      "           = FF FF FF etc.

(0) DB6 DATA TO DISK |      D       |      B       |      6       |      D       | etc.
    Data Bit Sequence| 1   1   0   1 | 1   0   1   1 | 0   1   1   0 | 1   1   0   1 | 1
    Recorded Flux-Rev  *   *)    (*   *)    (*   *)    (*   *)    (*   *)    (*   *)

(1) 6DB DATA TO DISK |      6       |      D       |      B       |      6       | etc.
    Data Bit Sequence| 0   1   1   0 | 1   1   0   1 | 1   0   1   1 | 0   1   1   0 | 1
    Recorded Flux-Rev    (*   *)    (*   *)    (*   *)    (*   *)    (*   *)    (*

(2) B6D DATA TO DISK |      B       |      6       |      D       |      B       | etc.
    Data Bit Sequence| 1   0   1   1 | 0   1   1   0 | 1   1   0   1 | 1   0   1   1 | 0
    Recorded Flux-Rev  *    (*   *)    (*   *)    (*   *)    (*   *)    (*   *)

(5) 00  DATA TO DISK |      0       |      0       |      0       |      0       | etc.
    Data Bit Sequence| 0   0   0   0 | 0   0   0   0 | 0   0   0   0 | 0   0   0   0 | 0
    Recorded Flux-Rev *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *

(8) FF  DATA TO DISK |      F       |      F       |      F       |      F       | etc.
    Data Bit Sequence| 1   1   1   1 | 1   1   1   1 | 1   1   1   1 | 1   1   1   1 | 1
    Recorded Flux-Rev *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *

    The pattern number (#) is the argument sent to the "Pagefill" ML routine.
    * indicates flux reversal points where direction of magnetization changes.
```

interfering with the new data when you try to read it.

Believe it or not, you could have media which is "too hot" for your drive. Your drive may be able to write these diskettes but not fully overwrite them.

The purpose of this discussion is to show that to test the media (and drive), you need to give it a sequence of writes, reads and seeks that will attempt to induce errors from the causes mentioned above; your actual operating modes will respect no orderly sequence of reads and writes.

Test Sequences

Figure 4 shows the sequence of writes and reads used in *DISKTEST*. It shows that each write is done in two passes, first by writing even tracks (or odd) while stepping in one direction and then by writing the odd (or even) tracks while stepping in the opposite direction. This is done to induce problems that would result from positioning errors. Not only that, but the pattern being written on

alternate tracks is different to induce problems from adjacent track interference.

Reads are also done in two passes following the odd/even scenario, except we always approach a track when reading from the opposite direction that we approached it when it was written. If this isn't fiendish enough, the rotating worst-case pattern test overwrites each track three times, each time with a different pattern and with the head approaching the track from the opposite direction than on the prior write.

If your drive and diskette media passes this test (and thankfully most do), then you have a high confidence that both are OK. If you run into problems which are widespread (occurring at many places on the disk), then it is more than likely that something is wrong with your drive or you have the wrong type of media for your drive. Media errors are random and generally not widespread.

Needless to say, the sources for errors

mentioned above are beyond the ability of most of us to fix, so a visit to your local repair center is called for if you cannot isolate the problem to the media.

Programming Techniques

Three machine language subroutines are used in the program to perform functions which were significant bottlenecks when implemented in BASIC. They are embedded in the BASIC program (lines 290, 295 and 300) in Hex program strings (P$). These contain the literal object code the assembler produced.

The string is defined and then a start address (X1) is sent to a subroutine (190) which pokes the string into memory. The same string name is reused on subsequent loads because this is done only once, and there is no reason to have a lot of string space tied up forever with these routines. At the end of the loads the program string is shortened by defining it to be of no length (""). Please check and recheck the entry of these strings.
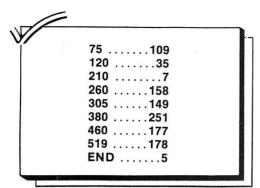
The first routine (Line 290) is too short to list, it is simply JSR [$C000] (Call *DSKCON*), followed by RTS. The second, *Pagefill* is described in Listing 2 and is in program Line 295. It puts the write patterns in the disk I/O buffer. The third is *Grantest* (Listing 3), which does an operation on all nine sectors of a granule and is in Line 300. These subroutines bring the run time down from over 30 minutes to a more reasonable level.

Editor's Note: *At the last minute, we discovered that the* Pagefill *and* Grantest *source code listings were inadvertently left out. They will appear in next month's* RAINBOW. *They do not affect running the main program.*

If you decide to study the program flow, here are a few tips. The array G(69) that stores the granule quality indicators does not always correlate with addressable logical granules. It does at the beginning, but before the test sequence the array is opened up in the middle to provide space for two pseuso-granules that the directory track occupies. At this point the array is considered to represent physical granules. Later, the array is closed up again (eliminating the directory granules), so the first 68 elements again correlate with the logical granules of the disk.

DISKTEST.UTL is offered for the personal use of readers of RAINBOW Magazine. I hope the program is useful and enhances your enjoyment of your CoCo system.

```
Figure 4        READ, WRITE & POSITIONING SEQUENCES

                            PATTERN & SEEK DIRECTION
  CYCLE#   OPERATION      EVEN TRACKS    ODD TRACKS
--------------------------------------------------------
  INTERLEAVED ROTATING WORST-CASE PATTERN TEST
--------------------------------------------------------
   1 (*)     WRITE          DB6  IN
             WRITE                        6DB OUT
             Read track 0 only; to set up for next
             READ                          IN
             READ           OUT
--------------------------------------------------------
   2 (*)     WRITE                        B6D  IN
             WRITE          6DB OUT
             READ           IN
             READ                         OUT
--------------------------------------------------------
   3 (*)     WRITE          B6D  IN
             WRITE                        DB6 OUT
             Read track 0 only; to set up for next
             READ                          IN
             READ           OUT
--------------------------------------------------------
  INTERLEAVED 1,0 TEST
--------------------------------------------------------
   4 (1)     WRITE          FF  IN
             WRITE                        00 OUT
             Read track 0 only; to set up for next
             READ                          IN
             READ           OUT
--------------------------------------------------------
   5 (2)     WRITE                        FF  IN
             WRITE          00 OUT
             READ           IN
             READ                         OUT
--------------------------------------------------------
   6 (3)     WRITE        Directory track only, FF
                      (a housekeeping cleanup pass)
--------------------------------------------------------
    Patterns DB6, 6DB, B6D, FF & 00 are described
    in Figure 2.
    (#) are cycle numbers shown in the short test,
    steps (*) are not done in the short test.
```

The listing:

```
10 'DISKTEST.UTL 1.1 (C) 1983 BY
CHARLES C. ZIMMER  101 AUSTIN RD
       SUDBURY, MA 01776
15 CLEAR90
20 BC=3:CLSBC
25 GOSUB180:GOTO 280
30 'i/o sub------------
35 POKEPP,O1:POKEPP+1,D1:POKEPP+
2,T1:POKEPP+3,S1:POKEPP+4,4:POKE
PP+5,0:X=USR0(0):EC=PEEK(PP+6):R
ETURN
40 'gran to disp sub----------
45 IFX>33THENEG=X+2ELSEG=X
50 IFG(X)=&HFF THENM$="   "ELSEM$
=RIGHT$(STR$(X),2)
55 PL=69+32*INT((G-24*INT(G/24))
/2)+10*INT(G/24)+(G-2*INT(G/2))*
3
60 IFG(X)<>&HE9 THENPRINT@PL,USI
NG"%%";M$;:RETURNELSEFORX2=0TO1:
POKE&H0400+PL+X2,ASC(MID$(M$,X2+
1,1)):NEXTX2:RETURN
65 'inkey sub------------
70 K$=INKEY$:IFK$=""THEN70ELSERE
TURN
75 'pause sub------------
80 PRINT@I1,M1$;:GOSUB70:GOSUB17
0:RETURN
85 'defect list form sub------
90 CLSBC:PRINT@3,"MAP OF DEFECTI
VE GRANULES:";:X3=0:FORX1=34TO54
STEP10:PRINT@X1,"TK/GR:GR";:FORX
2=1TO12:PRINT@X1+32*X2,USING"##"
;X3;:PRINT"/   :   ";:X3=X3+1:NEXT
X2,X1:PRINT@438,STRING$(8,F$);:P
RINT@239,"(DIR)";:RETURN
95 'list defects sub----------
100 GOSUB90:FORX=0TO67:IFG(X)=&H
B9 ORG(X)=&H99 ORG(X)=&HE9 THENG
OSUB45:NEXT:RETURNELSENEXT:RETUR
N
105 'declare defects sub--------
110 PRINT@I3,M3$;"DEFECT ENTRY P
OINT       ";:PRINT@I1,"<C>ONTINU
E   <E>NTER DEFECTS   ";:PLAYA$
115 GOSUB70:IFK$="C"THENRETURNEL
SEIFK$="E"THENGOSUB170:GOTO120EL
SEPLAYE$:GOTO115
120 PRINT@I3,STRING$(30,32);:PRI
NT@I1,"<I#>INSERT <D#>DELETE <C>
ONTIN";:PLAYA$:PRINT@I3,"";:INPU
T"  ";I$:PRINT@I3+30,F$;:L$=LEFT$
(I$,1):IFL$="C"THEN110ELSEIFL$="
D"ORL$="I"THEN125ELSEGOSUB155:GO
TO120
125 X1=LEN(I$)-1:FORX2=1TOX1:IFM
ID$(I$,X2+1,1)<"0"ORMID$(I$,X2+1
,1)>"9"THENX2=X1:GOSUB155:NEXT:G
OTO120ELSENEXT:I=VAL(RIGHT$(I$,X
1)):IFI<0ORI>67THENGOSUB155:GOTO
120
130 IFG(I)=&HB9 ORG(I)=&HE9 THEN
PRINT@I3,"CANNOT OVERRIDE TESTED
 RESULTS";:PRINT@I1,M1$;:PLAYE$E
LSE140
135 GOSUB70:GOTO120
140 IFL$="I"THENG(I)=&H99 ELSEG(
I)=&HFF
145 X=I:GOSUB45:GOTO120
150 'invalid resp sub----------
155 PRINT@I3,"INVALID ENTRY
```
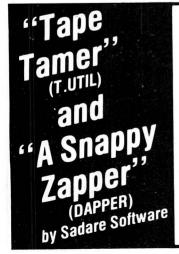
```
160 GOSUB70:RETURN
165 'msg clear sub-------------
170 PRINT@I3,C$;F$;F$;C$;:RETURN
175 'logo sub-------------
180 PRINT@134,"* D I S K   T E S
T *";:PRINT@224,"COPYRIGHT (C) 1
983 BY C C ZIMMER";:RETURN
185 'ml load sub-------------
190 FORX=0TOLEN(P$)/2-1:POKEX1+X
,VAL("&H"+MID$(P$,1+2*X,2)):NEXT
X:RETURN
195 'print question sub--------
200 PRINT@I1,"<P>FOR PRINTOUT  <
OTHER>CONTIN";:PLAYA$
205 GOSUB70:IFK$<>"P"THENP=0:RET
URNELSEP=1:PRINT@I1,"ENTER DATE
 <YYMMDD>                ";:PRINT@I3,S
TRING$(30,32);:PRINT@I3,"";:PLAY
A$:INPUTD$:PRINT@I3+30,F$;:RETUR
N
210 'diskette # sub------------
215 PRINT@I1,"ENTER DISKETTE NUM
BER <###>    ";:PRINT@I3,STRING$(
30,32);:PRINT@I3,"";:PLAYA$:INPU
TDN:LN$=LEFT$(LN$,8)+RIGHT$("000
"+RIGHT$(STR$(DN),LEN(STR$(DN))-
```

```
1),3):GOSUB170:RETURN
220 'print id sub-------------
225 POKE149,0:POKE150,18:POKE155
,96:PRINT#-2,CHR$(28);CHR$(27);C
HR$(48);
230 PRINT#-2,USING"% %d#%% %
% ";RIGHT$(LN$,3),D1,T$,D$;:RETU
RN
234 'print log sub-------------
235 X2=0:FORX=0TO67:IFG(X)=&HB9
THENPRINT#-2,"-";ELSEIFG(X)=&H99
 THENPRINT#-2,CHR$(34);ELSEIFG(X
)=&HE9 THENPRINT#-2,".";ELSEPRIN
T#-2,USING"#";X2;
240 X2=X2+1:IFX2=10THENX2=0:PRIN
T#-2,CHR$(27);CHR$(74);CHR$(28);
CHR$(31);CHR$(27);"T";:PRINT#-2,
USING"#";X/10;:PRINT#-2,CHR$(27)
;"I";CHR$(28);CHR$(27);CHR$(75);
:NEXTELSENEXT
245 PRINT#-2,"":RETURN
250 'drive # sub--------------
255 PRINT@I1,"ENTER DRIVE NUMBER
 (0-3)           ";:PRINT@I3,STRING$(
30,32);:PRINT@I3,"";:PLAYA$:INPU
TD1:GOSUB170:RETURN
260 'test type sub-------------
265 PRINT@I1,"<S>HORT TEST   <OTH
ER>FULL TEST";:PLAYA$
270 GOSUB70:IFK$="S"THENT$="Sh":
RETURNELSET$="Lg":RETURN
275 'initialization.............
...
280 I1=481:I3=449:FC=127+16*BC:B
$=CHR$(128):C$=STRING$(30,FC):F$
=CHR$(FC):PP=256*PEEK(&HC006)+PE
EK(&HC007):DIMG(69):DEFUSR0=&H0E
00:DEFUSR1=&H0E0A:DEFUSR2=&H0E86
:B=&H0400
285 LN$="DEFLOG   ":A$="V31;05;L3
5;T50;EP3EP3E":E$="V31;03;L35;T5
0;EP3EP3E"
290 P$="AD9FC00439":X1=&H0E00:GO
SUB190
295 P$="347FBDB3ED338D002D33CB10
AE8D00236F8D00211F31A68D001B8155
270EEC81EDA1A684A7A06C8D000B20E8
A684A7A4357F39040000DB6DB6DB6D00
0000FFFFFF":X1=&H0E0A:GOSUB190
300 P$="343FBDB3EDBEC006A7028609
3D4C5CE7031F02AD9FC0004E606271 0E6
02C1112605CC0002200ECC000120091F
20810926DCCC0000BDB4F4353F39":X1
=&H0E86:GOSUB190:P$=""
305 M1$="<ANY> TO CONTINUE
       ":M2$="<ANY> TO RESTART O
R QUIT          "
310 PRINT@I1,M1$;
315 PRINT@I3,"PLEASE TURN UP VOL
```

```
UME ON TV    ";:FORX=1TO150:NEXT:
PLAYE$:K$=INKEY$:IFK$=""THEN320E
LSEGOSUB170:GOSUB255:GOSUB265:GO
SUB200:GOTO350
320 PRINT@I3,"please";B$;"turn";
B$;"up";B$;"volume";B$;"on";B$;"
tv";B$;B$;B$;:FORX=1TO150:NEXT:P
LAYA$:GOTO315
325 'dir error abort point.....
330 PRINT@I3,"DIR. TRACK ERROR-C
AN'T CONTIN.";:IFP=1THENGOSUB225
:PRINT#-2,"DIR. TRACK ERROR-CANN
OT CONTINUE-DISK NOT USABLE OR U
NFORMATTED":PLAYE$ELSEPLAYE$
335 'restart point............
340 PRINT@I1,M2$;:GOSUB70:CLSBC
345 'start point..............
350 GOSUB180:PRINT@I1,"<C>OMMENC
E TEST    <Q>UIT        ";:PRINT@I3
,"INSERT OR CHANGE DISKETTES
";:PLAYA$
355 GOSUB70:IFK$="C"THENGOSUB170
:GOSUB215:GOTO365ELSEIFK$="Q"THE
NGOSUB170:PRINT@I3-33,"";:ENDELS
EPLAYE$:GOTO355
360 'file check................
365 GOSUB170:PRINT@357,"OPERATIO
N: FILE CHECK ";:T1=17:S1=2:01=2
```

```
:GOSUB35:IFEC<>0THEN330ELSE370
370 X1=1:FORX2=0TO67:G(X2)=PEEK(
B+X2):IFG(X2)<>&HFF THENX1=0
375 NEXT:IFX1=1THENPRINT@I3,"NO
FILES        ";:GOTO390
380 PRINT@I3,"files present    ";
385 'directory check..........
390 T1=17:S1=3:01=2:GOSUB35:IFEC
<>0THEN330
395 'log check.................
400 LF=1:FORX2=0TO07:IFASC(MID$(L
N$,X2+1,1))<>PEEK(B+X2)THENLF=0
405 NEXTX2:IFLF=1THENPRINT@I3+15
,"LOG PRESENT      ";ELSEPRINT@I3+
15,"NO DEFECT LOG   ";
410 PRINT@I1,"<C>ONTINUE   <OTHER
>ABORT TEST ";:PLAYA$
415 GOSUB70:IFK$="C"THENGOSUB170
:GOTO425ELSECLSBC:GOTO350
420 'log array prep...........
425 IFLF=0THEN435ELSEX1=PEEK(B+1
3)
430 X2=G(X1):G(X1)=&HB9:IFX2>=&H
C0 AND X2<=&HC9 THEN435ELSEIFX2<
=67THENX1=X2:GOTO430ELSEPRINT@I3
,"ERROR IN LOG - WILL TEST ALL
";:PLAYE$:LF=0:GOSUB80
435 FORX2=0TO67:IFLF=0THENX1=&HF
```

```
F:GOTO440ELSEIFG(X2)=&HB9 THENX1
=&HB9:GOTO440ELSEX1=&HFF
440 G(X2)=X1:NEXT
445 'first declare point.......
450 GOSUB100:M3$="first ":GOSUB1
10
455 'test sequence............
460 DATA0,34,2,0,3,33,1,-2,1,3,0
,0,1,0,0,1,33,2,1,2,34,0,-2,0,2,
1,33,2,2,3,34,0,-2,1,3,0,34,2,1,
2,33,1,-2,2,2,0,34,2,2,3,33,1,-2
,0,3,0,0,1,0,0,1,33,2,0,2,34,0,-
2,2,2
465 DATA0,34,2,8,3,33,1,-2,5,3,0
,0,1,0,0,1,33,2,5,2,34,0,-2,8,2,
1,33,2,8,3,34,0,-2,5,3,0,34,2,5,
2,33,1,-2,8,2
470 DATA17,17,1,8,3
475 'DATA0,3,1,0,2
480 DATA99,0,0,0,0
485 CLSBC:X1=.75:RESTORE:FORX=67
TO34STEP-1:G(X+2)=G(X):NEXT:DE=0
:G(34)=&HFF:G(35)=&HFF
490 IFT$="Sh"THENFORX=1TO70:READ
OP:NEXTX
495 READ TI,TE,ST,PA,OP:IFDE=1TH
EN565ELSEIFTI=99THENFORX=36TO69:
G(X-2)=G(X):NEXT:GOTO555ELSEIFOP
```

```
=3THENX=USR1(PA):GOTO500ELSEIFOP
=0THENO1=2:T1=TI:S1=1:GOSUB35:GO
TO495
500 O1=OP:X1=X1+.25:PRINT@295,"
    CYCLE:       ";:PRINT@327,"
    TRACK:       ";:PRINT@359,"
  granule:       ";:PRINT@391,"
OPERATION:       ";:PRINT@423,ST
RING$(18,32);
505 POKEPP,O1
510 FORT1=TI TO TE STEP ST:PRINT
@307,USING"##";INT(X1);:PRINT@33
9,USING"##";T1;:IFOP=2THENPRINT@
403,"READ ";ELSEPRINT@403,"WRITE
";
515 FORX2=0TOO1:PG=2*T1+X2
520 G1=2*T1+X2:IFT1>17THENG1=G1-
2
525 IFT1=17THENPRINT@371,"DIR";E
LSEPRINT@370,USING"### ";G1;
530 IFG(PG)=&HB9 ORG(PG)=&H99 OR
G(PG)=&HE9 THEN545
535 PRINT@423,STRING$(16,32);:X3
=256*T1+X2:EC=USR2(X3):IFEC=0THE
N545ELSEIFEC=2THENX2=1:T1=TE:DE=
1:GOTO540ELSEPRINT@423,"     BAD
GRANULE    ";:PLAYE$
540 G(PG)=&HE9
545 NEXTX2:NEXTT1:GOTO495
550 'final declare point.......
555 GOSUB100:M3$="final ":GOSUB1
10:CLSBC
560 'format allow table........
565 IFDE=1THEN330ELSEX=USR1(8):X
3=0:X2=0:X1=0
570 FORLG=0TO67:IFG(LG)=&H99 ORG
(LG)=&HB9 ORG(LG)=&HE9 THEN575EL
SE585
575 IFX3<>0THEN POKEB+X1,LG ELSE
X2=LG
580 X1=LG:X3=X3+1
585 NEXTLG:IFX3=0THEN590ELSEPOKE
B+X1,&HC9
590 T1=17:S1=2:O1=3:GOSUB35:FORX
4=1TO200:NEXTX4:IFEC<>0THEN330
595 'format dir entry.........
600 X=USR1(8):FORX=0TO10:POKEB+X
,ASC(MID$(LN$,X+1,1)):NEXT:POKEB
+11,1:POKEB+12,0:POKEB+13,X2:POK
EB+14,1:POKEB+15,0:IFX3=0THENPOK
EB,0
605 T1=17:S1=3:O1=3:GOSUB35:FORX
4=1TO200:NEXTX4:IFEC<>0THEN330
610 GOSUB90:PRINT@I3,"test cplt
- FINAL DEFECT LIST ";:FORX=0TO6
7:IFG(X)<>&HFF THENGOSUB45
615 NEXTX
620 IFP=1THENGOSUB225:GOSUB235
625 PLAYA$:GOTO340
```

A Simple Text Processor

By Ashok Basargekar

One of my favorite hobbies is to improve the Color Computer software written by others in my favorite RAINBOW magazine, give it a personal touch and enjoy the results. I remember Mr. Lewandowski's series of articles on the simple text handling program. I used to read the articles, enhance them to my satisfaction and wait for his next installment. After waiting for several months for him to give me some hints on the *EDIT* feature of his text handling program, I decided to take on this task myself.

Before going into the *EDIT* feature, I would like to present a complete face lift that I have given to the other subroutines of the text handler.

The first six lines of my assembly lan-

(Ashok Basargekar, a civil engineer in California, has developed an interest in computer science as a hobby. He has developed some programs in civil engineering subjects on CP/M. He received his Bachelor's degree in civil engineering from Baroda University in India and further courses in civil engineering and computer languages at California State University in Long Beach, Calif., Polytechnic University, Pomona and Santa Ana College in Santa Ana, Calif.)

guage source code define the ROM routines I will be using. The next 14 lines are the direct page addresses that I will be using to store my constants and variables. I may use a portion of the direct

> *"The Compose mode allows you to compose a new text, or to append a typed or loaded text from a tape or disk. Words will not wrap around to the next line while typing, but they will be properly moved to the next line at the time of printing on a paper."*

page; that's what the *Getting Started with Color BASIC* manual says! The START of my program uses the auto key repeat feature, published by Roger Schrag in his article on "Super Patched EDTASM". At START1 I release the alpha lock so I start my text processor with lowercase letters. In WIPE, I clear all the text buffer and then branch to FIN1 for my new menu. I beg your pardon, Mr. Lewandowski, I have used my name instead of yours, in the MES1. Instead of using LINPUT routine for

text handling, I have made it character-oriented in CONT for continue. I thought that the original PAPER routine was very primitive, so I changed it to give me the top of the form, left margin, line width and line spacing selections. First I take the characters up to the line width and go back to the nearest place where I can break a word before going to the next line. The *CLOAD, CSAVE, LOAD* and *SAVE* routines are the gifts of Roger Schrag from his disk and tape I/O routines. Before I go to the LINPUT for filename, I lock the alpha lock, so that the filename is always in capital letters. The EXIT routine also does the same thing. Finally I come to my *EDIT* routine for some comments.

Here I have used the same memory locations that I used to store the constants of PAPER routine in the direct page. SCL is used to store the text buffer address that will equate to the top left corner of the video screen. MARGIN stores the text buffer address that equates to the bottom right corner of the video screen. These addresses are revised as soon as the Y register (cursor pointer) goes beyond $400-$5FF range. Before bringing the next portion of the text for editing, all the previous buffer area is revised to match the screen

buffer. The COPY routine brings a copy of a portion of text in video screen for editing and the REVISE routine sends the edited text from screen to the text buffer. The NXTPGE and PRVPGE routines change the SCL and MARGIN addresses of next page or previous page depending upon the cursor movement. The DELETE routine moves all the text one to the left when the CLEAR key is pressed. The INSERT routine moves all the text one to the right for making room for a character in the middle.

I have used Spectral Associates' *ULTRA 80C* for editing and assembling this program. Of course, you may use any other assembler you wish. Since I have installed the *Lower-Kit*, by Green Mountain Micro, in my CoCo, the entire text is very beautiful on the screen.

The entire machine language code resides from $E00 through $16D4 and for a 32K computer, you will have plenty of text buffer area from $16D5 through $7FFF. The program is completely position independent except the address table for the menu subroutines. The control keys and procedure in using my *Text Processor* are as follows:

Initialization

LOADM"TEXT PRO" and *EXEC* will access this program. You will get a complete menu of selection as follows:

1) COMPOSE

The Compose mode allows you to compose a new text, or to append a typed or loaded text from a tape or disk. Words will not wrap around to the next line while typing, but they will be properly moved to the next line at the time of printing on a paper. Any immediate mistakes can be corrected by moving the cursor backward, with the left arrow key. Once you exit this Composing mode, and return back for continuing the text, you will not be able to correct the previously typed text with the left arrow key. You will need to go to the Edit mode for this purpose. While composing the text, do not press the ENTER key unless you want to go to the next line for a new paragraph. Pressing ENTER will provide a hard carriage return when printing the text on a printer. To exit the Composing mode, simply hit the BREAK key. You will return back to the main menu of selections.

2) EDIT

The text in the Edit mode appears slightly different from that in the Composing mode. You will see a red block at the places you have pressed the ENTER key, for providing a hard carriage return for a new paragraph. The up, down,

right and left arrow keys will move the cursor anywhere in the text, while in the Edit mode. The CLEAR key will delete one character at a time. The SHIFT-CLEAR keys will allow you to insert any text in the middle. The flashing cursor will disappear when you are in the Insert mode. You will return back to the Edit mode by pressing the BREAK key. You will exit the Edit mode by pressing the BREAK key again. The text can also be appended at the end while you are in the Insert mode. To revise the text in the Edit mode, simply write new text over the existing text.

". . .Simply enter the desired printing specifications for total line width, left margin and line spacing. Your text will be printed on the paper according to your specifications."

3) CLOAD

This selection will allow you to load a text from a cassette tape. The text can be loaded at the end of any typed or other-loaded text, allowing you to merge two or more texts.

4) DLOAD

This selection will allow you to load any text from a disk. You will be asked to enter a filename. The filename must be the entire name including the extension. If the filename is not found, or if the file is on a bad disk, you will receive an error message number. If so, simply press any key to go back to the main menu. Refer to Table 1 for the type of error.

5) PRINT

The underlining codes are presently set for the Brother Correctronic 50 typewriter. The Baud rate is set at 1200. Simply enter the desired printing specifications for total line width, left margin and line spacing. Your text will be printed on the paper according to your specifications. The paper will advance to the new page after printing 60 lines. Therefore, adjust the paper so that three blank lines are left at the top. This will provide three blank lines at the bottom. To change the printer Baud rate and printable lines per page or to change the underlining codes, you will need the following corrections to the software before executing the program.

POKE &HF74, msb : POKE &HF75, lsb of Baud rate constants.
POKE &H100D, n where n = printable lines per page.
POKE &H1016, m where m = blank lines at top and bottom of page.
POKE &H102C, 27 : POKE &H1031, 45 for start of underlining codes for Brother.
POKE &H1037, 27 : POKE &H103C, 82 for end of underlining codes for Brother.
POKE &H102C, 32 : POKE &H1031, 15 for start of underlining codes of LP VIII
POKE &H1037, 14 : POKE &H103C, 32 for end of underlining codes of LP VIII

6) CSAVE

This routine will allow you to save the text on a cassette tape.

7) DSAVE

This subroutine will allow you to save the text on a disk. You will be asked for a filename. It must be up to eight characters in length with an extension up to three characters. If an extension is not specified, none will be assumed. Therefore, give a filename like: *TEXT/DAT* or *TEXT.TXT*, etc.

The codes for the error messages while reading or writing text from or to the disk are as follows:

TABLE 1

CODE TYPE OF ERROR

CODE	TYPE OF ERROR
19	File already open
20	Bad device or drive number
21	I/O error
22	FM error
23	File not open
24	Input past end of line
27	File not found
29	Disk full
30	Out of buffer space
31	Disk write protected
32	Bad filename
33	Bad file structure
37	Verification error

8) EXIT

This will exit to BASIC. You will lose all the text with this selection. Therefore, make sure that the text is saved on the tape or disk prior to selecting EXIT.

Happy text processing! If you have any questions or suggestions regarding my text processor please drop a line with a SASE to Ashok Basargekar, 1423 North Cleveland Street, Orange, CA 92667, (714) 639-3996.

The listing:

```
              00010 *****************************        0E0C 20 62    00380      BRA START1
              00020 * A SIMPLE TEXT PROCESSOR    *        0E0E B6 0982  00390 NMI LDA $982
              00030 * BY ASHOK BASARGEKAR.       *        0E11 27 5C    00400      BEQ REPOUT
              00040 * 1423 NORTH CLEVELAND STREET,*       0E13 BE 0983  00410      LDX $983
              00050 * ORANGE, CA. 92667.         *        0E16 AF 6A    00420      STX $0A,S
              00060 *****************************        0E18 7F 0982  00430      CLR $982
              00061 *                                     0E1B 3B       00440      RTI
              00070 * MAJOR ROM ROUTINES USED BY THIS PROGRAM.  0E1C B6 FF03  00450 IRQ LDA $FF03
              00071 *                                     0E1F 2A 4E    00460      BPL REPOUT
A928          00080 CLS    EQU $A928 Clear screen.        0E21 B6 FF02  00470      LDA $FF02
A30A          00090 SCREEN EQU $A30A Print on screen.     0E24 B6 0985  00480      LDA $985
A393          00100 LINPUT EQU $A393 Line input.          0E27 27 10    00490      BEQ REPEAT
A2BF          00110 PRNTR  EQU $A2BF Print on printer.    0E29 7A 0985  00500      DEC $985
A1C1          00120 INKYS  EQU $A1C1 INKEY$               0E2C 26 0B    00510      BNE REPEAT
A027          00130 QUIT   EQU $A027 Back to Basic.       0E2E B6 0986  00520      LDA $986
A7D3          00140 DELAY  EQU $A7D3 Delay until X=0      0E31 84 B0    00530      ANDA #$B0
              00141 *                                     0E33 B7 0986  00540      STA $986
              00142 * Constants & variables stored in Direct Page.  0E36 B7 FF40  00550      STA $FF40
              00143 *                                     0E39 8E 0152  00560 REPEAT LDX #$152
0000          00150 KCLEAR EQU $0 Auto key repeat         0E3C A6 80    00570 RP1 LDA ,X+
0001          00160 KHOLD  EQU $1 constants.              0E3E 81 FF    00580      CMPA #$FF
0002          00170 BUFST  EQU $2 Start of text buffer addtess.  0E40 26 13    00590      BNE RP2
0004          00180 BUFEN  EQU $4 End of text buffer address.  0E42 8C 015A  00600      CMPX #$15A
0006          00190 SCL    EQU $6 Start of current line.  0E45 26 F5    00610      BNE RP1
0008          00200 MARGIN EQU $8 Left margin.            0E47 0C 00    00620      INC <KCLEAR
0009          00210 LW     EQU $9 Line width.             0E49 96 00    00630      LDA <KCLEAR
000A          00220 CLW    EQU $A Current line width.      0E4B 81 06    00640      CMPA #$6
000B          00230 SPACE  EQU $0B Line spacing.          0E4D 26 20    00650      BNE REPOUT
000C          00240 LCP    EQU $0C Line counter of page.  0E4F 0F 00    00660      CLR <KCLEAR
000D          00250 LENGTH EQU $0D Length of filename.     0E51 0F 01    00670      CLR <KHOLD
000E          00260 DSAVE  EQU $0E Tape/disk error vector.  0E53 20 1A    00680      BRA REPOUT
0011          00270 STACK  EQU $11 Tape/disk stack pointer.  0E55 0C 01    00690 RP2 INC <KHOLD
              00280 *                                     0E57 96 01    00700      LDA <KHOLD
0E00          00290        ORG $E00                       0E59 81 1E    00710      CMPA #$1E
              00300 *                                     0E5B 26 12    00720      BNE REPOUT
              00310 * Following interrupt service routine is similar to one  0E5D 80 03    00730      SUBA #$3
              00320 * in Rainbow Sept 83, page 77         0E5F 97 01    00740      STA <KHOLD
              00330 *                                     0E61 8E 0152  00750      LDX #$152
0E00 30 8C 0B 00340 START  LEAX <NMI,PCR                  0E64 A6 84    00760 RP3 LDA ,X
0E03 BF 010A  00350        STX $10A                       0E66 8A 3F    00770      ORA #$3F
0E06 30 8C 13 00360        LEAX <IRQ,PCR                  0E68 A7 80    00780      STA ,X+
0E09 BF 010D  00370        STX $10D                       0E6A 8C 015A  00790      CMPX #$15A
                                                          0E6D 26 F5    00800      BNE RP3
```

```
0E6F 3B          00810 REPOUT  RTI
                 00820 *
                 00830 * Entry to the main program with alpha lock released and
                 00840 * all text buffer cleared.
                 00850 *
0E70 7F   011A   00860 START1  CLR $11A
0E73 31   8D 0857  00870        LEAY BUFF,PCR
0E77 109F 02     00880         STY <BUFST
0E7A 109F 04     00890         STY <BUFEN
0E7D 86   00     00900         LDA #0
0E7F A7   A0     00910 WIPE    STA ,Y+
0E81 109C 25     00920         CMPY <$25 Top of RAM reached?
0E84 26   F9     00930         BNE WIPE
0E86 20   70     00940         BRA FIN1
                 00950 *
                 00960 * Print on screen routine.
                 00970 * Printing continues until a zero byte is reached.
                 00980 *
0E88 A6   80     00990 PRINT   LDA ,X+
0E8A 27   05     01000         BEQ DONE
0E8C BD   A30A   01010         JSR SCREEN
0E8F 20   F7     01020         BRA PRINT
0E91 39          01030 DONE    RTS
                 01040 *
                 01050 * Routine to continue with the text one character at a time
                 01060 * at the end of previous text.
                 01070 *
0E92 109E 04     01080 CONT    LDY <BUFEN
0E95 34   20     01090         PSHS Y
                 01100 *
                 01110 * Make sure that the flashing cursor does not go below
                 01111 * $400 the top left corner of video screen.
                 01120 *
0E97 9E   88     01140 FLASH   LDX <$88
0E99 8C   0400   01150         CMPX #$0400
0E9C 24   04     01160         BHS J1
0E9E 0C   89     01170         INC <$89
0EA0 20   F5     01180         BRA FLASH
                 01181 *
                 01182 * Alternately place a black ($80) and green ($8F) cursor
                 01183 * until a key is pressed.
                 01184 *
0EA2 B6   80     01190 J1      LDA #$80 Get a black cursor.
0EA4 8D   45     01200         BSR KBSCAN
0EA6 26   06     01210         BNE J2 Go to J2 if key pressed.
0EA8 86   8F     01220         LDA #$8F Wipe cursor with green.
0EAA 8D   3F     01230         BSR KBSCAN
0EAC 27   E9     01240         BEQ FLASH Zero means no key pressed.
                 01241 *
                 01242 * Place a character on screen until BREAK is presed.
                 01243 *
0EAE 81   03     01250 J2      CMPA #$03 BREAK?
0EB0 26   0B     01260         BNE J3
0EB2 86   00     01270         LDA #$0
0EB4 35   20     01280         PULS Y
0EB6 A7   A4     01290         STA ,Y
0EB8 109F 04     01300         STY <BUFEN
0EBB 20   3B     01310         BRA FIN1 Go to main menu routine.
                 01311 * If Back Space key is pressed, J4 makes it sure that
                 01312 * Y reg. is >=BUFEN of previously typed or loaded text.
                 01313 * J5 revises the text buffer address pointer and echoes
                 01314 * back space to screen. J6 ignores CLEAR key.
0EBD 81   08     01320 J3      CMPA #$8
0EBF 26   1B     01330         BNE J6
0EC1 86   8F     01340         LDA #$8F
0EC3 A7   9F 0088  01350       STA [$88]
0EC7 35   20     01360         PULS Y
0EC9 31   3F     01370         LEAY -1,Y
0ECB 109C 04     01380 J4      CMPY <BUFEN
0ECE 24   04     01390         BHS J5
0ED0 31   21     01400         LEAY 1,Y
0ED2 20   F7     01410         BRA J4
0ED4 86   00     01420 J5      LDA #$0
0ED6 A7   A4     01430         STA ,Y
0ED8 34   20     01440         PSHS Y
0EDA 86   08     01450         LDA #$8
0EDC BD   A30A   01460 J6      JSR SCREEN
0EDF 81   0C     01470         CMPA #$0C
0EE1 23   B4     01480         BLS FLASH
0EE3 35   20     01490         PULS Y
0EE5 A7   A0     01500         STA ,Y+
0EE7 34   20     01510         PSHS Y
0EE9 20   AC     01520         BRA FLASH
                 01530 *
                 01540 * This routine scans key board for a press. Returns zero
                 01550 * if none pressed.
                 01551 *
0EEB A7   9F 0088  01560 KBSCAN STA [$88]
0EEF 8D   A1C1    01570 J7      JSR INKYS
0EF2 26   03     01580         BNE J8
0EF4 5A          01590         DECB

0EF5 26   F8     01600         BNE J7
0EF7 39          01610 J8      RTS
                 01620 *
                 01630 * Main menu selection routine.
                 01640 *
0EF8 BD   A928   01650 FIN1    JSR CLS
0EFB 30   8D 067E  01660        LEAX MES1,PCR
0EFF 8D   87     01670         BSR PRINT
0F01 BD   A1C1   01680 WAIT    JSR INKYS
0F04 27   FB     01690         BEQ WAIT
0F06 80   31     01700         SUBA #$31
0F08 25   F7     01710         BLO WAIT
0F0A 81   08     01720         CMPA #$8
0F0C 24   F3     01730         BHS WAIT
0F0E 48          01740         ASLA
0F0F 8E   0F19   01750         LDX #MENU
0F12 AE   86     01760         LDX A,X
                 01770 * X now points to the absolute address of jump
0F14 BF   0F2A   01780         STX BRANCH
0F17 20   10     01790         BRA JUMP
                 01800 *
                 01810 * Table of address of different routines.
                 01820 *
0F19 0F2C        01830 MENU    FDB REST
0F1B 130D        01840         FDB EDIT
0F1D 104F        01850         FDB CLOAD
0F1F 11BD        01860         FDB LOAD
0F21 0F3B        01870         FDB PAPER
0F23 1173        01880         FDB SAVE
0F25 1173        01890         FDB SAVE
0F27 116E        01900         FDB EXIT
0F29 7E          01910 JUMP    FCB #7E
0F2A 0000        01920 BRANCH  FDB 0
                 01930 *
                 01940 * This routine prints all the text until end and goes
                 01950 * for continuation.
                 01960 *
0F2C BD   A928   01970 REST    JSR CLS
0F2F 9E   02     01980         LDX <BUFST
0F31 17   FF54   01990         LBSR PRINT
0F34 30   1F     02000         LEAX -1,X
0F36 9F   04     02010         STX <BUFEN
0F38 16   FF57   02020         LBRA CONT
                 02030 *
                 02040 * This routine gets the user specifications for printing
                 02050 * on printer and stores in the direct page.
                 02060 * The location SCL is used for temporary storage of each
                 02070 * user input.
                 02080 *
0F3B BD   A928   02090 PAPER   JSR CLS
0F3E 0F   0C     02100         CLR <LCP
0F40 30   8D 070B  02110       LEAX MES4,PCR
0F44 17   FF41   02120         LBSR PRINT
0F47 8D   31     02130         BSR SPECS
0F49 96   06     02140         LDA <SCL
0F4B 97   09     02150         STA <LW
0F4D 17   FF38   02160         LBSR PRINT
0F50 8D   28     02170         BSR SPECS
0F52 96   06     02180         LDA <SCL
0F54 97   08     02190         STA <MARGIN
0F56 96   09     02200         LDA <LW
0F58 90   08     02210         SUBA <MARGIN
0F5A 97   09     02220         STA <LW
0F5C 17   FF29   02230         LBSR PRINT
0F5F 8D   19     02240         BSR SPECS
0F61 96   06     02250         LDA <SCL
0F63 97   0B     02260         STA <SPACE
0F65 17   FF20   02270         LBSR PRINT
0F68 BD   A1C1   02280 L1      JSR INKYS
0F6B 81   03     02290         CMPA #$03
0F6D 27   89     02300         BEQ FIN1
0F6F 81   0D     02310         CMPA #$0D
0F71 26   F5     02320         BNE L1
0F73 8E   0029   02330         LDX #$0029  Baud Rate = 1200
0F76 9F   95     02340         STX <$95
0F78 20   33     02350         BRA GO
                 02360 *
                 02370 * This subroutine gets the user input of specifications,
                 02380 * converts from decimal to Hex number and returns in SCL
                 02390 *
0F7A 0F   06     02400 SPECS   CLR <SCL
0F7C 86   80     02410 L0      LDA #$80
0F7E 17   FF6A   02420         LBSR KBSCAN
0F81 26   07     02430         BNE L2
0F83 86   8F     02440         LDA #$8F
0F85 17   FF63   02450         LBSR KBSCAN
0F88 27   F2     02460         BEQ L0
0F8A 81   0D     02470 L2      CMPA #$0D
0F8C 26   01     02480         BNE L3
0F8E 39          02490         RTS
```

```
ØF8F 81  3Ø    Ø25ØØ L3    CMPA #$3Ø
ØF91 25  E9    Ø251Ø       BLO LØ
ØF93 81  39    Ø252Ø       CMPA #$39
ØF95 22  E5    Ø253Ø       BHI LØ
ØF97 BD  A3ØA  Ø254Ø       JSR SCREEN
ØF9A 8Ø  3Ø    Ø255Ø       SUBA #$3Ø
ØF9C ØD  Ø6    Ø256Ø       TST <SCL
ØF9E 26  Ø4    Ø257Ø       BNE L4
ØFAØ 97  Ø6    Ø258Ø L6    STA <SCL
ØFA2 2Ø  D8    Ø259Ø       BRA LØ
ØFA4 C6  ØA    Ø26ØØ L4    LDB #$ØA
ØFA6 9B  Ø6    Ø261Ø L5    ADDA <SCL
ØFA8 5A        Ø262Ø       DECB
ØFA9 26  FB    Ø263Ø       BNE L5
ØFAB 2Ø  F3    Ø264Ø       BRA L6
               Ø265Ø * This is the main entry for printing text on printer.
               Ø267Ø *
ØFAD 9E  Ø2    Ø27ØØ 6Ø    LDX <BUFST
               Ø27Ø1 *
               Ø27Ø2 * Start address of current line to be printed is stored
               Ø27Ø3 * at SCL, no. of characters that can be printed within
               Ø27Ø4 * selected line width and margin is determined and is
               Ø27Ø5 * stored at CLW.
               Ø27Ø6 *
ØFAF 9F  Ø6    Ø271Ø LP99  STX <SCL
ØFB1 5F        Ø272Ø       CLRB
ØFB2 A6  8Ø    Ø273Ø LP1   LDA ,X+
ØFB4 27  28    Ø274Ø       BEQ STORE
ØFB6 81  ØD    Ø275Ø       CMPA #$ØD    CR?
ØFB8 27  24    Ø276Ø       BEQ STORE
ØFBA 5C        Ø277Ø       INCB
ØFBB D1  Ø9    Ø278Ø       CMPB <LW
ØFBD 26  F3    Ø279Ø       BNE LP1
ØFBF 3Ø  1F    Ø28ØØ       LEAX -1,X
ØFC1 81  2Ø    Ø281Ø LP2   CMPA #$2Ø    SPACE?
ØFC3 27  19    Ø282Ø       BEQ STORE
ØFC5 81  2E    Ø283Ø       CMPA #$2E    PERIOD?
ØFC7 27  15    Ø284Ø       BEQ STORE
ØFC9 81  21    Ø285Ø       CMPA #$21
ØFCB 27  11    Ø286Ø       BEQ STORE
ØFCD 81  3B    Ø287Ø       CMPA #$3B
ØFCF 27  ØD    Ø288Ø       BEQ STORE

ØFD1 81  2D    Ø289Ø       CMPA #$2D
ØFD3 27  Ø9    Ø29ØØ       BEQ STORE
ØFD5 81  3F    Ø291Ø       CMPA #$3F
ØFD7 27  Ø5    Ø292Ø       BEQ STORE
ØFD9 A6  82    Ø293Ø       LDA ,-X
ØFDB 5A        Ø294Ø       DECB
ØFDC 2Ø  E3    Ø295Ø       BRA LP2
ØFDE D7  ØA    Ø296Ø STORE STB <CLW
               Ø297Ø *
               Ø298Ø * Main routine for printing a line on printer.
               Ø299Ø *
ØFEØ C6  FE    Ø3ØØØ       LDB #$FE Device #-2
ØFE2 D7  6F    Ø3Ø1Ø       STB <$6F
ØFE4 9E  Ø6    Ø3Ø2Ø       LDX <SCL
               Ø3Ø21 * Print specified left margin if any.
ØFE6 D6  Ø8    Ø3Ø3Ø       LDB <MARGIN
ØFE8 27  Ø8    Ø3Ø4Ø       BEQ LP4
ØFEA 86  2Ø    Ø3Ø5Ø       LDA #$2Ø
ØFEC BD  A2BF  Ø3Ø6Ø LP3   JSR PRNTR
ØFEF 5A        Ø3Ø7Ø       DECB
ØFFØ 26  FA    Ø3Ø8Ø       BNE LP3
ØFF2 D6  ØA    Ø3Ø9Ø LP4   LDB <CLW
ØFF4 A6  8Ø    Ø31ØØ LP13  LDA ,X+
ØFF6 81  ØØ    Ø311Ø       CMPA #$ØØ
ØFF8 27  4B    Ø312Ø       BEQ LP5
ØFFA 81  ØD    Ø313Ø       CMPA #$ØD
ØFFC 26  25    Ø314Ø       BNE LP6
               Ø315Ø *
               Ø316Ø * This routine sends line feeds equal to spacing selected,
               Ø317Ø * after printing each line.
               Ø318Ø *
ØFFE D6  ØB    Ø319Ø LP14  LDB <SPACE
1ØØØ 26  Ø1    Ø32ØØ       BNE LP7
1ØØ2 5C        Ø321Ø       INCB
1ØØ3 86  ØD    Ø322Ø LP7   LDA #$ØD
1ØØ5 BD  A2BF  Ø323Ø       JSR PRNTR
1ØØ8 ØC  ØC    Ø324Ø       INC <LCP
1ØØA 96  ØC    Ø325Ø       LDA <LCP
1ØØC 81  3C    Ø326Ø       CMPA #$3C    6Ø LINES?
1ØØE 27  Ø5    Ø327Ø       BEQ LP8
1Ø1Ø 5A        Ø328Ø       DECB
1Ø11 26  FØ    Ø329Ø       BNE LP7
1Ø13 2Ø  9A    Ø33ØØ       BRA LP99
               Ø331Ø *
               Ø332Ø * This routine skips six lines after printing sixty lines
               Ø333Ø * on each page and goes to new page.
               Ø334Ø *
1Ø15 C6  Ø6    Ø335Ø LP8   LDB #6       6 BLANK LINES.
1Ø17 86  ØD    Ø336Ø LP1Ø  LDA #$ØD
1Ø19 BD  A2BF  Ø337Ø       JSR PRNTR
1Ø1C 5A        Ø338Ø       DECB
1Ø1D 26  F8    Ø339Ø       BNE LP1Ø
1Ø1F ØF  ØC    Ø34ØØ       CLR <LCP
1Ø21 2Ø  8C    Ø341Ø       BRA LP99
               Ø342Ø * This routine prints one character at a time on printer.
               Ø343Ø * Check is made for special printer commands for underlining.
               Ø344Ø *
1Ø23 81  2Ø    Ø345Ø LP6   CMPA #$2Ø
1Ø25 27  16    Ø346Ø       BEQ LP11
1Ø27 81  3C    Ø347Ø       CMPA #$3C    <?
1Ø29 26  Ø7    Ø348Ø       BNE LP12
1Ø2B 86  1B    Ø349Ø       LDA #$1B
1Ø2D BD  A2BF  Ø35ØØ       JSR PRNTR
1Ø3Ø 86  45    Ø351Ø       LDA #$45
1Ø32 81  3E    Ø352Ø LP12  CMPA #$3E    >?
1Ø34 26  Ø7    Ø353Ø       BNE LP11
1Ø36 86  1B    Ø354Ø       LDA #$1B
1Ø38 BD  A2BF  Ø355Ø       JSR PRNTR
1Ø3B 86  52    Ø356Ø       LDA #$52
1Ø3D BD  A2BF  Ø357Ø LP11  JSR PRNTR
1Ø4Ø 5A        Ø358Ø       DECB
1Ø41 26  B1    Ø359Ø       BNE LP13
1Ø43 2Ø  B9    Ø36ØØ       BRA LP14
               Ø361Ø *
               Ø362Ø * This routine sends final carriage return, changes device
               Ø363Ø * code to screen and returns to main menu.
               Ø364Ø *
1Ø45 86  ØD    Ø365Ø LP5   LDA #$ØD
1Ø47 BD  A2BF  Ø366Ø       JSR PRNTR
1Ø4A ØF  6F    Ø367Ø       CLR <$6F
1Ø4C 16  FEA9  Ø368Ø       LBRA FIN1
               Ø369Ø *
               Ø37ØØ * Load from cassette tape routine.
               Ø371Ø * BREAK key will abort routine and will return to main menu.
               Ø372Ø *
1Ø4F BD  A928  Ø373Ø CLOAD JSR CLS
1Ø52 3Ø  8D Ø667 Ø374Ø     LEAX MES5,PCR
1Ø56 17  FE2F  Ø375Ø       LBSR PRINT
1Ø59 BD  A1C1  Ø376Ø WAIT2 JSR INKYS
1Ø5C 27  FB    Ø377Ø       BEQ WAIT2
```

```
105E 81   03      03780          CMPA #$03
1060 1027 FE94    03790          LBEQ FIN1
                  03800 *
                  03810 * Tape load routine is similar to that in Oct.83 Rainbow
                  03820 * page 84
                  03830 *
1064 C6   FF      03840          LDB #$FF   Select motor on.
1066 17   00BB    03850          LBSR MOTOR
1069 1026 0286    03860          LBNE ERROR
106D AE   8D 049E 03870          LDX NONAME,PCR
1071 86   49      03880          LDA #$49    Select input from tape.
1073 C6   FF      03890          LDB #$FF    Select on screen.
1075 17   01C9    03900          LBSR COPEN
1078 1026 0277    03910          LBNE ERROR
107C 9E   04      03920          LDX <BUFEN
107E 17   0210    03930 LOOP4    LBSR CINPUT
1081 1026 026E    03940          LBNE ERROR
1085 A7   80      03950          STA ,X+
1087 4D          03960          TSTA
1088 26   F4      03970          BNE LOOP4
108A 30   1F      03980          LEAX -1,X
108C 9F   04      03990          STX <BUFEN
108E 17   01E0    04000          LBSR CCLOSE
1091 1026 025E    04010          LBNE ERROR
1095 C6   00      04020          LDB #$0    Select motor off.
1097 17   008A    04030          LBSR MOTOR
109A 1026 0255    04040          LBNE ERROR
109E 16   FE57    04050          LBRA FIN1
                  04060 *
                  04070 * Routine for user input of tape/disk filename.
                  04080 *
10A1 BD   A928    04090 NAME     JSR CLS
10A4 8E   02DD    04100          LDX #$2DD
10A7 CC   2055    04110          LDD #$2055
10AA A7   80      04120 LOOP2    STA ,X+
10AC 5A          04130          DECB
10AD 26   FB      04140          BNE LOOP2
10AF B6   011A    04150          LDA $11A
10B2 34   02      04160          PSHS A
10B4 86   FF      04170          LDA #$FF Set the alpha lock for
10B6 B7   011A    04180          STA $11A Capital letter filename.
10B9 30   8D 046E 04190 REDO     LEAX MES,PCR
10BD 17   FDC8    04200          LBSR PRINT
10C0 BD   A393    04210          JSR $A393   Get name.
10C3 D1   0D      04220          CMPB <LENGTH Valid length?
10C5 2E   F2      04230          BGT REDO Do it again if invalid.
10C7 35   04      04240          PULS B Reset the
10C9 F7   011A    04250          STB $11A alpha lock.
10CC 39          04260          RTS
                  04270 *
                  04280 * Routine to save text on cassette tape.
                  04290 * See Oct 83 Rainbow page 84
                  04300 *
10CD C6   09      04310 CSAVE    LDB #9
10CF D7   0D      04320          STB <LENGTH
10D1 8D   CE      04330          BSR NAME
10D3 30   8D 05E6 04340          LEAX MES5,PCR
10D7 17   FDAE    04350          LBSR PRINT
10DA BD   A1C1    04360 WAIT3    JSR $A1C1
10DD 27   FB      04370          BEQ WAIT3
10DF 81   03      04380          CMPA #$03
10E1 1027 FE13    04390          LBEQ FIN1
                  04400 * Main CSAVE routine.
10E5 C6   FF      04410          LDB #$FF   Select motor on.
10E7 8D   3B      04420          BSR MOTOR
10E9 1026 0286    04430          LBNE ERROR
10ED 8E   02DD    04440          LDX #$2DD   Point at name.
10F0 86   4F      04450          LDA #$4F   Select output to tape.
10F2 C6   FF      04460          LDB #$FF   Select on screen.
10F4 17   014A    04470          LBSR COPEN
10F7 1026 01FB    04480          LBNE ERROR
10FB 9E   02      04490          LDX <BUFST
10FD A6   80      04500 CLOOP    LDA ,X+   Read a character.
10FF 4D          04510          TSTA
1100 27   09      04520          BEQ SOUT
1102 17   017C    04530          LBSR CSTPRT
1105 1026 01EA    04540          LBNE ERROR
1109 20   F2      04550          BRA CLOOP
110B 17   0173    04560 SOUT     LBSR CSTPRT
110E 1026 01E1    04570          LBNE ERROR
1112 17   015C    04580          LBSR CCLOSE
1115 1026 01DA    04590          LBNE ERROR
1119 C6   00      04600          LDB #$0   Select motor off.
111B 8D   07      04610          BSR MOTOR
111D 1026 01D2    04620          LBNE ERROR
1121 16   FDD4    04630          LBRA FIN1
                  04640 *
                  04650 * This routine turns cassette motor on or off (B=0 : off)
                  04660 *
1124 17   01A7    04670 MOTOR    LBSR BEGIN
1127 5D          04680          TSTB
1128 26   06      04690          BNE MOTORN
112A BD   A7EB    04700          JSR $A7EB   Motor off.

112D 16   01C7    04710          LBRA L21
1130 BD   A7CA    04720 MOTORN   JSR $A7CA   Motor on.
1133 16   01C1    04730          LBRA L21
                  04740 *
                  04750 * Routine to process cassette file name.
                  04760 *
1136 D7   68      04770 CNAME    STB $68
1138 CE   01D1    04780          LDU #$1D1
113B 6F   C0      04790          CLR ,U+
113D C6   20      04800          LDB #$20
113F E7   C0      04810 CLEAR    STB ,U+
1141 1183 01DA    04820          CMPU #$1DA
1145 25   F8      04830          BLO CLEAR
1147 CE   01D2    04840          LDU #$1D2
114A E6   80      04850 PNAME    LDB ,X+
114C C1   20      04860          CMPB #$20
114E 25   0B      04870          BLO RETURN
1150 E7   C0      04880          STB ,U+
1152 7C   01D1    04890          INC $1D1
1155 1183 01DA    04900          CMPU #$1DA
1159 25   EF      04910          BLO PNAME
115B 39          04920 RETURN   RTS
                  04930 *
                  04940 * Abort save on tape/disk routines if text buffer is empty.
                  04950 *
115C BD   A928    04960 NOTXT    JSR CLS
115F 30   8D 040C 04970          LEAX ERMES,PCR
1163 17   FD22    04980          LBSR PRINT
1166 BD   A1C1    04990 WAIT4    JSR $A1C1
1169 27   FB      05000          BEQ WAIT4
116B 16   FD8A    05010          LBRA FIN1
                  05020 *
                  05030 * Exit to basic with a cold start restoring interrupts and
                  05040 * alpha lock.
                  05050 *
116E 0F   71      05060 EXIT     CLR <$71
1170 7E   A027    05070          JMP QUIT
                  05080 *
                  05090 * Save on tape/disk routines.
                  05100 *
1173 34   02      05110 SAVE     PSHS A
1175 DC   04      05120          LDD <BUFEN
1177 93   02      05130          SUBD <BUFST
1179 27   E1      05140          BEQ NOTXT
117B 35   02      05150          PULS A
117D 81   0A      05160          CMPA #10
```

```
117F 1027 FF4A  05170      LBEQ CSAVE
                05180 *
                05190 * Disk save routines. Refer to July 83 Rainbow page 71
                05200 *
1183 8D 2E      05210      BSR SETUP
1185 17 008D    05220      LBSR LABEL
1188 8E 02DD    05230      LDX #$2DD
118B 108E 01FF  05240      LDY #$1FF
118F B6 4F      05250      LDA #$4F
1191 C6 01      05260      LDB #$1
1193 17 00C9    05270      LBSR OPEN
1196 26 56      05280      BNE GOOFED
1198 30 8D 0532 05290      LEAX BUFF,PCR
119C A6 80      05300 WRITE LDA ,X+
119E C6 01      05310      LDB #$1
11A0 17 00E5    05320      LBSR DSKPRT
11A3 26 49      05330      BNE GOOFED
11A5 81 00      05340      CMPA #0
11A7 26 F3      05350      BNE WRITE
11A9 C6 01      05360      LDB #$1
11AB 17 00CA    05370      LBSR CLOSE
11AE 26 3E      05380      BNE GOOFED
11B0 16 FD45    05390      LBRA FIN1
                05400 *
                05410 * Routine to setup one buffer and verify on.
                05420 *
11B3 C6 01      05430 SETUP LDB #$1
11B5 8D 69      05440      BSR FILES
11B7 C6 01      05450      LDB #$1
11B9 17 00EB    05460      LBSR VERIFY
11BC 39         05470      RTS
                05480 *
                05490 * Routine to load a disk data file.
                05500 *
11BD 8D F4      05510 LOAD  BSR SETUP
11BF 8D 54      05520      BSR LABEL
11C1 8E 02DD    05530      LDX #$2DD
11C4 108E 01FF  05540      LDY #$1FF
11C8 86 49      05550      LDA #$49
11CA C6 01      05560      LDB #$1
11CC 17 0090    05570      LBSR OPEN
11CF 26 1D      05580      BNE GOOFED
11D1 9E 04      05590      LDX <BUFEN
11D3 C6 01      05600 READ  LDB #$1
11D5 17 00C4    05610      LBSR INPUT
11D8 26 14      05620      BNE GOOFED
11DA A7 80      05630      STA ,X+
11DC 81 00      05640      CMPA #0
11DE 26 F3      05650      BNE READ
11E0 30 1F      05660      LEAX -1,X
11E2 9F 04      05670      STX <BUFEN
11E4 C6 01      05680      LDB #$1
11E6 17 008F    05690      LBSR CLOSE
11E9 26 03      05700      BNE GOOFED
11EB 16 FD0A    05710      LBRA FIN1
11EE 34 04      05720 GOOFED PSHS B
11F0 BD A928    05730      JSR CLS
11F3 35 04      05740      PULS B
11F5 30 8D 0318 05750      LEAX ERRMSG,PCR
11F9 17 FC8C    05760      LBSR PRINT
11FC 86 2F      05770      LDA #$2F
11FE 4C         05780 ERR   INCA
11FF C0 0A      05790      SUBB #$0A
1201 24 FB      05800      BCC ERR
1203 CB 3A      05810      ADDB #$3A
1205 BD A30A    05820      JSR $A30A
1208 1F 98      05830      TFR B,A
120A BD A30A    05840      JSR $A30A
120D BD A1C1    05850 LL1   JSR INKYS
1210 27 FB      05860      BEQ LL1
1212 16 FCE3    05870      LBRA FIN1
                05880 *
                05890 * Routine to process disk filename.
                05900 *
1215 34 34      05910 LABEL PSHS B,X,Y
1217 C6 0D      05920      LDB #$0D
1219 D7 00      05930      STB <LENGTH
121B 17 FE83    05940      LBSR NAME
121E 35 B4      05950      PULS PC,B,X,Y
                05960 * Routine to setup disk system memory.
1220 17 00AB    05970 FILES LBSR BEGIN
1223 34 04      05980      PSHS B
1225 BD CA3B    05990      JSR $CA3B
1228 35 04      06000      PULS B
122A F7 095B    06010      STB $95B
122D CE 0928    06020      LDU #$928
1230 8E 0989    06030      LDX #$989
1233 6F 84      06040 DOBUF CLR ,X
1235 AF C1      06050      STX ,U++
1237 30 89 0119 06060      LEAX $119,X
123B 5A         06070      DECB
123C 22 F5      06080      BHI DOBUF
123E 16 00B6    06090      LBRA L21
                06100 * Routine to open cassete file.
1241 17 008A    06110 COPEN LBSR BEGIN
1244 17 FEEF    06120      LBSR CNAME
1247 81 49      06130      CMPA #$49
1249 27 07      06140      BEQ OPENI
124B 81 4F      06150      CMPA #$4F
124D 27 09      06160      BEQ OPENO
124F 7E A616    06170      JMP $A616
1252 BD A629    06180 OPENI JSR $A629
1255 16 009F    06190      LBRA L21
1258 4F         06200 OPENO CLRA
1259 BD A65B    06210      JSR $A65B
125C 16 0098    06220      LBRA L21
                06230 * Routine to open disk file.
125F 8D 6D      06240 OPEN  BSR BEGIN
1261 108F 0957  06250      STY $957
1265 34 06      06260      PSHS D
1267 8D 45      06270      BSR FNAME
1269 35 06      06280      PULS D
126B BD C468    06290      JSR $C468
126E 16 0086    06300      LBRA L21
                06310 * Routine to close cassete tape file.
1271 8D 5B      06320 CCLOSE BSR BEGIN
1273 BD A437    06330      JSR $A437
1276 20 7F      06340      BRA L21
                06350 * Routine to close disk file.
1278 8D 54      06360 CLOSE BSR BEGIN
127A D7 6F      06370      STB $6F
127C BD CA53    06380      JSR $CA53
127F 20 76      06390      BRA L21
                06400 * Routine to write on tape.
1281 8D 4B      06410 CSTPRT BSR BEGIN
1283 BD A290    06420      JSR $A290
1286 20 6F      06430      BRA L21
                06440 * Routine to write on disk.
1288 8D 44      06450 DSKPRT BSR BEGIN
128A D7 6F      06460      STB $6F
128C BD A282    06470      JSR $A282
128F 20 66      06480      BRA L21
                06490 * Routine to read tape file.
1291 8D 3B      06500 CINPUT BSR BEGIN
1293 0F 70      06510      CLR $70
1295 BD A17F    06520      JSR $A17F
1298 A7 60      06530      STA 0,S
129A 20 5B      06540      BRA L21
                06550 * Routine to read disk file.
129C 8D 30      06560 INPUT BSR BEGIN
129E D7 6F      06570      STB $6F
12A0 BD A176    06580      JSR $A176
12A3 A7 E4      06590      STA ,S
12A5 20 50      06600      BRA L21
                06610 * Routine to set verify on.
12A7 8D 25      06620 VERIFY BSR BEGIN
12A9 F7 0987    06630      STB $987
12AC 20 49      06640      BRA L21
                06650 * Routine to process disk filename.
12AE C6 FF      06660 FNAME LDB #$FF
12B0 5C         06670 GETLEN INCB
12B1 A6 85      06680      LDA B,X
12B3 81 20      06690      CMPA #$20
12B5 24 F9      06700      BCC GETLEN
12B7 6F E2      06710      CLR ,-S
12B9 B6 095A    06720      LDA $95A
12BC 97 EB      06730      STA $EB
12BE CE 094C    06740      LDU #$94C
12C1 86 20      06750      LDA #$20
12C3 A7 C0      06760 L22   STA ,U+
12C5 1183 0957  06770      CMPU #$957
12C9 26 F8      06780      BNE L22
12CB 7E C8A4    06790      JMP $C8A4
                06800 * Routine to prepare everything.(Registers,Error traps etc)
12CE 34 7A      06810 BEGIN PSHS X,Y,U,DP,A
12D0 4F         06820      CLRA
12D1 1F 8B      06830      TFR A,DP
12D3 B6 018E    06840      LDA $18E
12D6 FE 018F    06850      LDU $18F
12D9 97 0E      06860      STA <DSAVE
12DB DF 11      06870      STU <$11
12DD 86 7E      06880      LDA #$7E
12DF 33 8D 0010 06890      LEAU ERROR,PCR
12E3 B7 018E    06900      STA $18E
12E6 FF 018F    06910      STU $18F
12E9 A6 E4      06920      LDA ,S
12EB 10DF 11    06930      STS <STACK
12EE EE 66      06940      LDU 6,S
12F0 6E F8 08   06950      JMP [8,S]
                06960 * Pass through this routine if error.
12F3 54         06970 ERROR LSRB
12F4 5C         06980      INCB
12F5 20 03      06990      BRA L25
                07000 * Pass through this if no error.
12F7 5F         07010 L21   CLRB
12F8 20 00      07020      BRA L25
                07030 * Restore registers and return to caller.
12FA 96 0E      07040 L25   LDA <DSAVE
12FC DE 11      07050      LDU <$11
12FE B7 018E    07060      STA $18E
1301 FF 018F    07070      STU $18F
1304 10DE 11    07080      LDS <STACK
1307 35 7A      07090      PULS A,DP,U,Y,X
```

```
1309 32  62     07100 LEAS 2,S
130B 5D         07110 TSTB
130C 39         07120 RTS
                07130 * Edit routine uses video screen display area ($0400-$5FF)
                07140 * to display portions of text buffer for editing.
                07150 * Editor uses direct page addresses as follows:
                07160 * CLW : Cursor address upon entry to new screen page.
                07170 * SCL : Start of current text buffer address corresponding
                07180 * to top left corner of video screen.
                07190 * MARGIN : End of current text buffer address corresponding
                07200 * to bottom right corner of video screen.
130D 8E  0400   07210 EDIT LDX #$0400 First, the cursor pointer at top left corner.
1310 9F  0A     07220 STX <CLW
1312 DC  02     07230 EDIT1 LDD <BUFST
1314 DD  06     07240 NEWPGE STD <SCL
1316 C3  01FF   07250 ADDD #$1FF
1319 1093 04    07260 CMPD <BUFEN
131C 25  05     07270 BLO SKIP
131E DC  04     07280 LDD <BUFEN
1320 83  0001   07290 SUBD #1
1323 DD  08     07300 SKIP STD <MARGIN
1325 17  00B3   07310 LBSR COPY
1328 109E 0A    07320 LDY <CLW
                07330 * This routine waits for user to press a key. Y reg. points
                07340 * to the screen address of cursor location. The character
                07350 * and a black cursor ($80) are flashed alternately until
                07360 * a key is pressed.
132B E6  A4     07370 EDWAIT LDB ,Y
132D 34  04     07380 PSHS B Save character on stack.
132F 8E  0400   07390 LDX #$0400
1332 BD  A7D3   07400 JSR DELAY
1335 BD  A1C1   07410 JSR INKYS
1338 C6  80     07420 LDB #$80 Get a black cursor.
133A E7  A4     07430 STB ,Y Place it at cursor pointer.
133C 8E  0400   07440 LDX #$0400
133F BD  A7D3   07450 JSR DELAY
1342 35  04     07460 PULS B Get the character from stack.
1344 E7  A4     07470 STB ,Y Place it again at cursor pointer.
1346 81  00     07480 CMPA #0
1348 27  E1     07490 BEQ EDWAIT
134A 81  03     07500 CMPA #3 BREAK?
134C 26  06     07510 BNE SKIP0
                07520 * Always revise the text buffer to match screen before
                07530 * exiting routine.
134E 17  00AB   07540 LBSR REVISE
1351 16  FBA4   07550 LBRA FINI
                07560 * Check if any of the arrow keys is pressed.
                07570 * Revise cursor pointer if arrow key pressed.
                07580 * If cursor pointer goes beyond screen display area,
                07590 * go to next page or previous page.
1354 81  0A     07600 SKIP0 CMPA #$0A
1356 26  0D     07610 BNE SKIP1
1358 31  20     07620 LEAY 32,Y
135B 108C 05FF  07630 EDCHK CMPY #$5FF
135F 1022 00A7  07640 LBHI NXTPGE
1363 20  C6     07650 BRA EDWAIT
1365 81  09     07660 SKIP1 CMPA #$09
1367 26  04     07670 BNE SKIP2
1369 31  21     07680 LEAY 1,Y
136B 20  EE     07690 BRA EDCHK
136D 81  5E     07700 SKIP2 CMPA #$5E
136F 26  0D     07710 BNE SKIP3
1371 31  A8 E0  07720 LEAY -32,Y
1374 108C 0400  07730 EDCK CMPY #$0400
1378 1025 00A6  07740 LBLO PRVPGE
137C 20  AD     07750 BRA EDWAIT
137E 81  08     07760 SKIP3 CMPA #$08
1380 26  04     07770 BNE SKIP4
1382 31  3F     07780 LEAY -1,Y
1384 20  EE     07790 BRA EDCK
                07800 * CLEAR key will branch to DELETE routine and SHIFT CLEAR
                07810 * key will branch to INSERT routine.
1386 81  0C     07820 SKIP4 CMPA #$0C
1388 1027 00AE  07830 LBEQ DELETE
138C 81  5C     07840 CMPA #$5C
138E 1027 00FB  07850 LBEQ INSERT
                07851 * Place the edited character at cursor pointer.
1392 8D  0A     07860 BSR CHANGE
1394 A7  A0     07870 STA ,Y+
1396 108C 05FF  07880 CMPY #$5FF
139A 22  6E     07890 BHI NXTPGE
139C 20  8D     07900 BRA EDWAIT
                07910 * This routine changes the ASCII of character for
                07920 * screen printing.
139E 81  80     07930 CHANGE CMPA #$80
13A0 26  02     07940 BNE SK5
13A2 86  FF     07950 LDA #$FF
13A4 81  0D     07960 SK5 CMPA #$0D
13A6 26  02     07970 BNE SKIP5
13A8 86  BF     07980 LDA #$BF
13AA 81  60     07990 SKIP5 CMPA #$60
13AC 25  08     08000 BLO SKIP6
13AE 81  80     08010 CMPA #$80
13B0 24  04     08020 BHS SKIP6
13B2 80  60     08030 SUBA #$60
13B4 20  06     08040 BRA CHNG0K
13B6 81  40     08050 SKIP6 CMPA #$40
13B8 24  02     08060 BHS CHNG0K

13BA 8B  40     08070 ADDA #$40
13BC 39         08080 CHNG0K RTS
                08090 * This routine converts back the screen byte into ASCII
                08100 * character for placing it in the text buffer.
13BD 81  FF     08110 UNCHNG CMPA #$FF
13BF 26  03     08120 BNE NONULL
13C1 4F         08130 CLRA
13C2 20  16     08140 BRA UNOK
13C4 81  BF     08150 NONULL CMPA #$BF
13C6 26  04     08160 BNE SKIP7
13C8 86  0D     08170 LDA #$0D
13CA 20  0E     08180 BRA UNOK
13CC 81  40     08190 SKIP7 CMPA #$40
13CE 24  04     08200 BHS SKIP8
13D0 8B  60     08210 ADDA #$60
13D2 20  06     08220 BRA UNOK
13D4 81  60     08230 SKIP8 CMPA #$60
13D6 25  02     08240 BLO UNOK
13D8 80  40     08250 SUBA #$40
13DA 39         08260 UNOK RTS
                08270 * This routine brings a copy of portion of the text
                08280 * buffer area to the video screen.
13DB BD  A928   08290 COPY JSR CLS
13DE 108E 0400  08300 LDY #$0400
13E2 9E  06     08310 LDX <SCL
13E4 A6  80     08320 COP1 LDA ,X+
13E6 8D  B6     08330 BSR CHANGE
13E8 A7  A0     08340 COP2 STA ,Y+
13EA 9C  08     08350 CMPX <MARGIN
13EC 23  F6     08360 BLS COP1
13EE 108C 0600  08370 CMPY #$0600
13F2 27  04     08380 BEQ COPOUT
13F4 86  FF     08390 LDA #$FF
13F6 20  F0     08400 BRA COP2
13F8 39         08410 COPOUT RTS
                08420 * This subroutine takes the edited text from the screen
                08430 * area and places it back at the proper location in the
                08440 * text buffer area.
13F9 108E 0400  08450 REVISE LDY #$0400
13FD 9E  06     08460 LDX <SCL
13FF A6  A0     08470 REV1 LDA ,Y+
1401 8D  BA     08480 BSR UNCHNG
1403 A7  80     08490 STA ,X+
1405 9C  08     08500 CMPX <MARGIN
1407 23  F6     08510 BLS REV1
1409 39         08520 RTS
                08530 * If Y > $5FF cursor goes to top of next page in this
                08540 * routine. The text buffer is always revised to match
                08550 * screen buffer before going to next page.
140A 31  A9 FE00 08560 NXTPGE LEAY -512,Y
140E 109F 0A    08570 STY <CLW
1411 8D  E6     08580 BSR REVISE
1413 DC  08     08590 LDD <MARGIN
1415 C3  0001   08600 ADDD #1
1418 1093 04    08610 CMPD <BUFEN
141B 1027 FEF3  08620 LBEQ EDIT1
141F 16  FEF2   08630 LBRA NEWPGE
                08640 * If Y < $400 the cursor goes to bottom of previous page.
                08650 * The text buffer is always revised to match screen buffer
                08660 * before going to previous page.
1422 31  A9 0200 08670 PRVPGE LEAY $200,Y
1426 109F 0A    08680 STY <CLW
1429 8D  CE     08690 BSR REVISE
142B DC  06     08700 LDD <SCL
142D 83  0200   08710 SUBD #$200
1430 1093 02    08720 CMPD <BUFST
1433 1025 FEDB  08730 LBLO EDIT1
1437 16  FEDA   08740 LBRA NEWPGE
                08750 * Delete a character routine.
                08760 * One character at a time is deleted and the text on
                08770 * screen is moved one to the left upto the bottom right
                08780 * corner of screen. The next character from the text buffer
                08790 * area is brought to screen. All the characters in the
                08800 * text buffer are also moved one address down.
143A 1F  21     08810 DELETE TFR Y,X Get cursor pointer in X reg.
143C 30  01     08820 LEAX 1,X       X=X+1
143E 8C  0600   08830 CMPX #$0600 Is it beyond screen buffer?
1441 27  26     08840 BEQ DEL2 Go to DEL* if yes.
1443 A6  84     08850 LDA ,X Get the character.
1445 A7  82     08860 STA ,-X Place it to the left.
1447 81  FF     08870 CMPA #$FF End of the text?
1449 26  11     08880 BNE DEL1 Goto DEL1 if not.
                08881 * End of text means time to revise text buffer.
144B 9E  08     08890 LDX <MARGIN
144D 9F  04     08900 STX <BUFEN
144F 30  1F     08910 LEAX -1,X
1451 9F  08     08920 STX <MARGIN
1453 34  20     08930 PSHS Y Save Video screen cursor pointer.
1455 8D  A2     08940 BSR REVISE
1457 35  20     08950 PULS Y Get back the cursor pointer.
1459 16  FECF   08960 LBRA EDWAIT
                08961 * Keep on shifting characters to the left until end of
                08962 * video screen buffer.
145C 30  02     08970 DEL1 LEAX 2,X
145E 8C  0600   08980 CMPX #$0600
1461 27  06     08990 BEQ DEL2
1463 A6  84     09000 LDA ,X
```

```
1465 A7 82     09010         STA ,-X
1467 20 F3     09020         BRA DEL1
               09021 * Time to place first character from next text buffer
               09022 * area into bottom right corner of video screen buffer.
1469 9E 08     09030 DEL2    LDX <MARGIN
146B 30 01     09040         LEAX 1,X
146D A6 84     09050         LDA ,X
146F 27 12     09060         BEQ DEL4 Goto DEL4 if end of text.
1471 FF2A      09070         LBSR CHANGE
1474 B7 05FF   09080         STA $5FF
1477 30 1F     09090         LEAX -1,X
               09091 * Move all the text buffer characters one address down.
1479 30 02     09100 DEL3    LEAX 2,X
147B A6 84     09110         LDA ,X
147D 27 04     09120         BEQ DEL4
147F A7 82     09130         STA ,-X
1481 20 F6     09140         BRA DEL3
               09141 * shift last character one address down.
1483 A7 82     09150 DEL4    STA ,-X
1485 9F 04     09160         STX <BUFEN
1487 16 FEA1   09170         LBRA EDWAIT
               09180 * Inserting characters, one at a time in the middle.
               09190 * All the text after cursor location is moved one to the
               09200 * right to make room for new user input. The text in the
               09210 * text buffer area after the character at bottom right of
               09220 * screen is also moved up one address at the same time.
148A 1F 21     09230 INSERT  TFR Y,X Get cursor pointer in X reg.
               09231 * Wait for insert.
148C BD A1C1   09240 INWAIT  JSR INKYS
148F 27 FB     09250         BEQ INWAIT
1491 81 03     09260         CMPA #03 BREAK?
1493 1027 FE94 09270         LBEQ EDWAIT Exit insert mode.
1497 17 FF04   09280         LBSR CHANGE
149A 34 02     09290         PSHS A Save character to be inserted.
149C A6 80     09300         LDA ,X+ Get character at cursor pointer : X=X+1
149E 8C 0600   09310         CMPX #0600 Out of screen buffer?
14A1 27 2A     09320         BEQ INS2 Go to INS2 if yes.
14A3 E6 84     09330 INS0    LDB ,X Get next character in B reg.
14A5 A7 80     09340         STA ,X+ Place previous character to the right.
14A7 8C 0600   09350         CMPX #0600
14AA 27 1F     09360         BEQ INS1
14AC 1E 89     09370         EXG A,B Switch character from B to A
14AE 81 FF     09380         CMPA #FF End of text?
14B0 26 F1     09390         BNE INS0 Keep on moving characters to the right.
14B2 8C 05FF   09400         CMPX #$5FF End of screen buffer?
14B5 27 EC     09410         BEQ INS0
               09411 * When text buffer is smaller than screen buffer following
               09412 * routine is required.
14B7 34 10     09420         PSHS X Save cursor pointer.
14B9 9E 04     09430         LDX <BUFEN X=text buffer end pointer.
14BB 9F 08     09440         STX <MARGIN Store it to correspond bottom of screen buffer.
14BD 30 01     09450         LEAX 1,X X=X+1
14BF 9F 04     09460         STX <BUFEN Increase BUFEN
14C1 35 10     09470         PULS X Get cursor pointer again.
14C3 A7 84     09480         STA ,X Store character at cursor pointer.
14C5 35 02     09490         PULS A Get character to be inserted.
14C7 A7 A8     09500         STA ,Y+ Place it at cursor pointer. : Y=Y+1
14C9 20 BF     09510         BRA INSERT Branch to insert.
               09511 * Here when text buffer needs to be moved up one address.
14CB 1E 89     09520 INS1    EXG A,B Switch characters from B to A.
               09521 * Character from Bottom right corner of video screen
14CD 17 FEED   09530 INS2    LBSR UNCHNG
14D0 9E 08     09540         LDX <MARGIN
14D2 30 01     09550         LEAX 1,X
               09551 * This section moves the remaining text buffer one address up.
14D4 E6 84     09560 INS3    LDB ,X
14D6 A7 80     09570         STA ,X+
14D8 1E 89     09580         EXG A,B
14DA 81 00     09590         CMPA #0
14DC 26 F6     09600         BNE INS3
14DE A7 84     09610         STA ,X
14E0 9F 04     09620         STX <BUFEN
14E2 35 02     09630         PULS A Get the charater to be inserted.
14E4 A7 A8     09640         STA ,Y+ Place it at cursor pointer. : Y=Y+1
14E6 108C 05FF 09650         CMPY #$5FF Within screen buffer?
14EA 23 9E     09660         BLS INSERT Go insert more characters.
               09661 * Make sure to revise text buffer corresponding to the
               09662 * screen buffer
14EC 17 FF0A   09670         LBSR REVISE
               09671 * Insert to continue at top of video screen as new page.
14EF DC 08     09680         LDD <MARGIN
14F1 C3 0001   09690         ADDD #1
14F4 DD 06     09700         STD <SCL
14F6 C3 01FF   09710         ADDD #$1FF
14F9 1093 04   09720         CMPD <BUFEN
14FC 25 05     09730         BLO INS4
14FE DC 04     09740         LDD <BUFEN
1500 83 0001   09750         SUBD #1
               09751 * Insert continues here when next text buffer is smaller
               09752 * than video screen buffer.
1503 DD 08     09760 INS4    STD <MARGIN
1505 17 FED3   09770         LBSR COPY
1508 108E 0400 09780         LDY #$400
150C 16 FF7B   09790         LBRA INSERT
```

```
150F    0000    09800 NONAME  FDB 0
1511    3C      09810 ERRMSG  FCC "<Break> TO EXIT, ERROR #"
1529    0000    09820         FDB $0
152B    20      09830 MES     FCC "    LOAD/SAVE ROUTINE"
1541    0D      09840         FCB $0D
1542    20      09850         FCC "    PRESS break TO EXIT"
155A    0D0D    09860         FDB $0D0D
155C    20      09870         FCC " enter FILE NAME: "
156E    00      09880         FCB $0
156F    0D      09890 ERMES   FCB $0D
1570    42      09900         FCC "BUFFER EMPTY"
157C    00      09910         FCB $00
157D    20      09920 MES1    FCC *   A SIMPLE TEXT PROCESSOR*
1598    0D      09930         FCB $0D
1599    20      09940         FCC *     BY A.K. BASARGEKAR*
15B1    0D0D    09950         FDB $0D0D
15B3    20      09960         FCC * TEXT IN < > WILL BE UNDERLINED*
15D2    0D      09970         FCB $0D
15D3    20      09980         FCC * SELECT 1-8. HIT break FOR MENU*
15F2    0D0D    09990         FDB $0D0D
15F4    20      10000         FCC * 1 - COMPOSE*
1600    0D      10010         FCB $0D
1601    20      10020         FCC * 2 - EDIT*
160A    0D      10030         FCB $0D
160B    20      10040         FCC * 3 - CLOAD*
1615    0D      10050         FCB $0D
1616    20      10060         FCC * 4 - DLOAD*
1620    0D      10070         FCB $0D
1621    20      10080         FCC * 5 - PRINT*
162B    0D      10090         FCB $0D
162C    20      10100         FCC * 6 - CSAVE*
1636    0D      10110         FCB $0D
1637    20      10120         FCC * 7 - DSAVE*
1641    0D      10130         FCB $0D
1642    20      10140         FCC * 8 - EXIT*
164B    00      10150         FCB $0
164C    20      10160 MES4    FCC " TOTAL LINE WIDTH = "
1660    000D    10170         FDB $000D
1662    0D      10180         FCB $0D
1663    20      10190         FCC " LEFT MARGIN = "
1672    000D    10200         FDB $000D
1674    0D      10210         FCB $0D
1675    20      10220         FCC " LINE SPACING = "
1685    000D    10230         FDB $000D
1687    0D      10240         FCB $0D
1688    20      10250         FCC " TURN ON PRINTER AND enter"
16A2    0D0D    10260         FDB $0D0D
16A4    20      10270         FCC " OR PRESS break TO EXIT"
16BB    00      10280         FCB 0
16BC    00      10290         FCB 0
16BD    52      10300 MES5    FCC "READY CASSETTE?"
16CC    0D00    10310         FDB $0D00
16CE            10320 BUFF    RMB 0
        0E00    10330         END START
```

One-Liner Contest Winner . . .

Did you ever hear the old question about whether a penny doubled each day for a month is more than a million dollars? This program gives you the true scoop on your printer.

The listing:
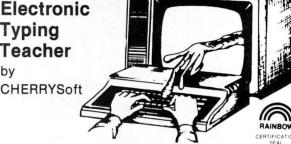
```
10 POKE150,42:PRINT#-2,"KING OF
THE ONE-LINERS, PENNY DBLR";CHR$
(13);"SEND PRIZE TO Lee Veal, 88
09 Linda Vista, Rowlett, TX 7508
8":A=.01:PRINT#-2:PRINT#-2,TAB(1
0)"AMOUNT";" DAY":FOR X=1TO31:PR
INT#-2,USING"$$###,###,###.##";A
;:PRINT#-2," ";X:A=A+A:NEXT
```

Lee Veal
Rowlett, TX

(For this winning one-liner contest entry, the author has been sent copies of both *The Rainbow Book Of Adventure* and its companion *Rainbow Adventure Tape*.)

This is the final part in a series on diskette file organization

The Diskette
Directories
Handler System

By Marvin E. Swan

The complete "Diskette Directories Handler" system is being presented in three parts. You must *RUN* program *DIR1* before *RUNning* this month's program *DIR3*.

You can refer to RAINBOW two months ago for Part 1 and a full overview of all seven comprehensive reports generated

by the *Handler* system and an explanation of program *DIR1*. You can refer to last month's RAINBOW for an explanation of program *DIR2*. To refresh your memory, here is the cross reference of all programs and reports and their *RUN* sequence:

Consolidated Directories, Report #6

This month, report #6 is generated by program *DIR3*. The purpose of report #6 is to show all programs and data files cataloged to your PC system, sequenced by name, extension, format, type, granule size, byte size and diskette name. You can look up a particular program/data filename and see where it resides and how many multiple versions you may have saved to more than one diskette. An asterisk appears under the "FLAG" column when you have different byte-sized BASIC or machine language programs of the same name, allowing you to determine the correct version to *RUN* or *EXECUTE*. This report is the nucleus of the *Handler* system, which you will reference more often than others. You can cross reference all programs/data files in your PC system giving you the ability to purge and manage your entire diskette library.

Program Name	Optional Report	Report Title	Sequenced By:
DIR0	no	DOCUMENTATION	
DIR1	yes	#1 EXTR. & LOAD DIR	your own diskette filing
DIR1	yes	#2 DISKETTE NAMES	diskette name
DIR1	yes	#3 DISKETTE USAGE	number of bytes used
DIR1	yes	#4 DISK JACKET LABELS	your own diskette filing
DIR2	yes	#5 DISK LABELS (gum)	your own diskette filing
DIR3	no	#6 CONSOLIDATION	prog/size/diskette name
DIR3	no	#7 SUMMARY (stats)	

Disks Directory Summary, Report #7

This report is generated by program *DIR3*. The purpose of report #7 is to show statistical and percentage totals of all your diskettes, granules, bytes, programs and data files in your system. The following statistics and percentages will print:

* Total diskettes
* Bytes/granules used, not used, final total
  (Note that percentage between granules and bytes do not match because granules are in increments of 2,304 bytes each and bytes are absolute.)
* Total BASIC, assembly and machine language programs
* Total data files
* Total multiple program versions with different byte size
* Total diskettes with disk read errors encountered
* Total reports you requested from *Handler* system.

The purpose of *DIR3* is to print the *Consolidated Directories* report #6 and the statistical *Disks Directory Summary* report #7. *DIR3* is automatically *RUN* by program *DIR2* if you previously selected optional report #5, otherwise it is automatically *RUN* by program *DIR1*. The entire catalog file is sorted in RAM for the consolidation report. The sort is unique because it passes the catalog file six times, allowing for faster sorting on smaller amounts of data giving the advantage of being able to sort up to 2,100 records from disk. I strongly urge you to replace the BASIC bubble sort with a machine language sort. I purchased one advertised through RAINBOW that literally sorts in seconds. When *DIR2* is complete, be sure to back up the catalog file to an archival diskette. The following BASIC line numbers contain Epson brand printer codes that you may change to satisfy your printer brand requirements and a high speed poke for the bubble sort: 0490, 0740 and 0820.

Program *DIR0*

The purpose of *DIR0* is to give you a quick blurb on how programs *DIR1*, *DIR2* and *DIR3* integrate. It gives you a record layout of the catalog file, BASIC line numbers within all programs for high speed pokes and Epson print codes and a host of other useful information. You do not have to *RUN* this program to process the *Handler* system. *DIR0* simply prints a quick and dirty explanation of the *Handler* system when you can't find your RAINBOW for reference.

The Final Saga

I sincerely hope you enjoy using the *Handler* system. My friends and I have been using it successfully for about 18 months. I am putting together a subsystem to the *Handler* system which allows you to add and delete directory information to the catalog file without having to reprocess all your diskettes again when you want to regenerate the seven reports. I am hoping RAINBOW will share it with you in the near future.

3-D Filing Cabinet With 'Windows'

If you have not been keying in th four programs, you may *CLOADM* them off the RAINBOW ON TAPE or mail a formatted diskette with $8.50 to Marv's Computer Room, 17323 40th Avenue South, Seattle, WA 98188, and I will mail them to you. In addition to the "complete" *Handler* system, I will include a program that displays up to 15 graphic *PMODE 4* windows, showing a filing cabinet with drawers that open and shut, and printer paper images with text coming out of the drawers and hanging in front of the cabinet. The *Diskette Directories Handler* system can be *RUN* from one of the four drawers. The other three drawers will surprise you. Windows are commonly used on the more expensive business micros.

Listing 1:

```
10 '[DIR3]
20 'PROGRAM 3 OF 3
30 'COPYRIGHT 1984 BY MARV SWAN
40 '
50 CLS:PRINT
60 VERIFYON:GOTO930
70 CLEAR19500
80 DD$="disks directory":L=66:T$
="FILE "
90 FT$(0)="Program Basic":FT$(1)
="Data File":FT$(2)="Program Mac
h L":FT$(3)="Program Assy L":DIM
D$(450):G$(1)="' ' TO 'B'":G$(2)
="'C' TO 'E'":G$(3)="'F' TO 'L'"
:G$(4)="'M' TO 'Q'":G$(5)="'R' T
O 'S'":G$(6)="'T' TO 'Z'":G=1
100 GOSUB690:SOUND10,3:PRINT"mak
e sure THAT THE DISK WORK    FIL
E CREATED BY PROGRAM DIR1     IS
IN DRIVE 0":PRINT@224,"type FILE
 NAME OF 1-8 CHARACTERSWITHOUT E
XTENSION":PRINT"  ........"
110 PRINT@288,"";:INPUTDI$
120 IFLEN(DI$)<1 OR LEN(DI$)>8 G
OTO100
130 I=INSTR(DI$,"/"):IFI<>0 GOTO
100
140 I=INSTR(DI$,"."):IFI<>0 GOTO
100
150 FORX=3TO11
160 DSKI$0,17,X,A$,B$:F=INSTR(B$
,DI$):IFF>0THENFT=FT+1
170 F=INSTR(A$,DI$):IFF>0THENFT=
FT+1
180 NEXT
190 IFFT=0THENSOUND10,3:GOSUB690
:PRINT@128,"'";DI$;"' NOT FOUND
ON DISKETTE":PRINT"press ENTER T
O retype FILE NAME":GOSUB700:GOT
0100 ELSEDI$=DI$+".DIR"
200 OPEN"I",#1,DI$:INPUT#1,DA$
210 GOSUB680:PRINT" reading inpu
t directory file ";:PRINT@129,T$
```

```
"NAME:";::PRINT@140,DI$;
220 PRINT@161,T$"DATE:";::PRINT@1
72,DA$;:PRINT@193,T$"PASS:";:I$=
STR$(G):I$=RIGHT$(I$,1):PRINT@20
4,I$" OF 6";:PRINT@225,"REC # IN
:";::PRINT@236,"0";:PRINT@257,"DA
TA IN:";::PRINT@268,"NONE";:PRINT
@289,"SEARCHING:";::PRINT@300,"PR
EFIX "G$(G);
230 PRINT@352," records selected
 for sort ";::PRINT@385,"REC # OU
T:";::PRINT@396,"0";:PRINT@417,"D
ATA OUT:";::PRINT@428,"NONE";
240 P=P+1
250 INPUT#1,R$:TT=TT+1:I$=STR$(T
T-1):I$=RIGHT$(I$,LEN(I$)-1):PRI
NT@236,I$S$;
260 IFR$="TOTALS"THENSOUND10,3:D
$(P)=R$:S$=STR$(TT-1):S$=" OF "+
RIGHT$(S$,LEN(S$)-1):IFP<>1GOTO4
30ELSE720
270 PRINT@268,LEFT$(R$,8)" "MID$
(R$,9,3);
280 A$=LEFT$(R$,1)
290 IFG=1ANDA$<"C"GOTO410
300 IFG=1GOTO250
310 IFG=2ANDA$>"B"ANDA$<"F"GOTO4
10
320 IFG=2GOTO250
330 IFG=3ANDA$>"E"ANDA$<"M"GOTO4
10
340 IFG=3GOTO250
350 IFG=4ANDA$>"L"ANDA$<"R"GOTO4
10
360 IFG=4GOTO250
370 IFG=5ANDA$>"Q"ANDA$<"T"GOTO4
10
380 IFG=5GOTO250
390 IFA$>"S"GOTO410
400 GOTO250
410 D$(P)=LEFT$(R$,41)
420 I$=STR$(P):I$=RIGHT$(I$,LEN(
I$)-1):PRINT@396,I$;:PRINT@428,L
EFT$(R$,8)" "MID$(R$,9,3);:GOTO2
40
430 SOUND10,3:GOSUB820:GOSUB680:
I$=STR$(G):PRINT" PRINTING REPOR
T #6, PASS "RIGHT$(I$,1)"..."
440 IFG=1 THEN PT=INT((TT/57)+.9
99999999):PK=0
450 JK=0
460 IFL>60GOSUB650 ELSEGOTO530
470 PK=PK+1:IFPK<10THENTB=46ELSE
TB=45
480 IFPT>9THENTB=TB-1
490 PRINT#-2,TAB(5)DA$;TAB(16)CH
```

```
R$(14);"consolidated directories
";CHR$(20);TAB(TB)"page";PK;"of"
;PT
500 PRINT#-2:PRINT#-2,TAB(5)"nam
e     ext  format   type";TAB(40)
"grans    bytes  diskette name";T
AB(75)"flag"
510 PRINT#-2,TAB(5)"====    ===
  ======  ====";TAB(40) "=====
=====  =============";TAB(75) "==
=="
520 L=L+4
530 JK=JK+1:R$=D$(JK)
540 IFMID$(R$,12,1)="+"THENPRINT
#-2,TAB(5)"+++++++++ +++    Fatal D
isk Read Error    00        0   ";M
ID$(R$,22,20):GOTO610
550 IF MID$(R$,12,1)="B" THEN FR
MT$="Binary" ELSE FRMT$="Ascii"
560 TYPE=VAL(MID$(R$,13,1))
570 BYTES=VAL(MID$(R$,16,6))
580 IF HR$=LEFT$(R$,13) AND HBYT
E$<>MID$(R$,16,6) THEN ERR$="*":
ERR=ERR+1:GOTO600 ELSE ERR$=""
590 HR$=LEFT$(R$,13):HBYTE$=MID$
(R$,16,6)
600 PRINT#-2,TAB(5)LEFT$(R$,8);T
AB(14)MID$(R$,9,3);TAB(19)FRMT$;
TAB(27)FT$(TYPE);TAB(43)MID$(R$,
14,2);:PRINT#-2,TAB(46);:PRINT#-
2,USING"###,###";BYTES;:PRINT#-2
,TAB(55)RIGHT$(R$,20);:IF ERR$="
" THEN PRINT#-2 ELSE PRINT#-2,TA
B(78)ERR$
610 L=L+1
620 IFJK=P-1GOTO720
630 IFL>60GOTO460
640 GOTO530
650 IFL=66GOTO670
660 PRINT#-2:L=L+1:IFL<>66GOTO66
0
670 L=0:RETURN
680 SOUND10,3
690 CLS2:PRINT@1,"PRINT CONSOLID
ATED DIRECTORIES";:PRINT@41,"PRO
GRAM 3 OF 3";:PRINT@96,"";:RETUR
N
700 I$=INKEY$:IFI$=""GOTO700ELSE
RETURN
710 FORS=1TO500:NEXTS:RETURN
720 IFG<>6THENCLOSE#1:G=G+1:P=0:
TT=0:GOTO200
730 GOSUB690:PRINT" PRINTING REP
ORT #7..."
740 GOSUB650:PRINT#-2,DA$;TAB(15
);CHR$(14);"disks directory summ
ary";CHR$(20);TAB(47)"page 1 of
1"
750 INPUT#1,T$,T
```

```
760 IFT$="1"THENPRINT#-2:GOTO750
770 IFT$="END"THENCLOSE#1:PRINT@
416,"";:END
780 IFRIGHT$(T$,7)="flagged"THEN
T=ERR
790 J=40-LEN(T$):PRINT#-2,TAB(9)
:PRINT#-2,USING"#,###,###,###";T
;:PRINT#-2,STRING$(J,".");:PRINT
#-2,T$
800 IF(RIGHT$(T$,7)="flagged"AND
ERR>0)THENI=(ERR/TT)*100:I=I+.25
:I=INT(I):PRINT#-2,TAB(9):PRINT#
-2,USING"#,###,###,###";I;:PRINT
#-2,"............percentage ver
sions flagged"
810 GOTO750
820 POKE65495,0 '<===== note
830 GOSUB690
840 FORX=1TOP-1:P1=INT((X/P)*100
):PRINT@96,P1;"% SORTED"
850 FORY=X TOP-1:IFD$(Y)<D$(X)TH
ENS1$=D$(X):D$(X)=D$(Y):D$(Y)=S1
$
860 NEXTY:NEXT X
870 PRINT@96," 100 % SORTED"
880 PRINT@160,P-1"RECORDS SORTED
"
890 IFP-1>50THENPRINT@224," WHEW
!!"
900 FORX=1TO800:NEXT
910 POKE65494,0
920 RETURN
930 PMODE0:PCLEAR1:GOTO70
```

```
290 ....... 41
580 ....... 184
760 ....... 248
END ...... 255
```

Listing 2:
```
10 '[DIR0]
20 '
30 'DOCUMENTATION FOR PROGRAMS
     DIR1, DIR2 AND DIR3
40 'RUN THIS PROGRAM AND A HARD
     COPY WILL PRINT EXPLAINING
     THE 3 PROGRAMS
50 '
60 '
70 '
80 VERIFYON:CLEAR500
90 DATA"  D o c u m e n t a t i o
   n"
100 DATA" "
110 DATA"          f o r"
120 DATA" "
130 DATA"Diskette Directories Ha
ndler"
140 DATA"Programs DIR1, DIR2 and
```
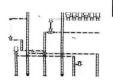
```
DIR3"
150 DATA"Copyright 1984 by Marv
Swan"
160 DATA" "
170 DATA" "
180 DATA"The Diskette DIrectorie
s Handler system consists of 3 p
rograms"
190 DATA"that generate all your
diskette directories information
 onto"
200 DATA"7 comprehensive reports
"
210 DATA" "
220 DATA" "
230 DATA"Minimum computer system
 requirements:"
240 DATA"  * 32K OF RAM"
250 DATA"  * EXTENDED DISK BASIC
260 DATA"  * 1 DISK DRIVE"
270 DATA"  * 1 TAPE DRIVE"
280 DATA"  * 1 PRINTER"
290 DATA" "
300 DATA"Program DIR1 explanatio
n:"
310 DATA"  * READ ALL DISKETTE D
IRECTORIES IN YOUR SYSTEM"
320 DATA"      AND EXTRACT INFORMA
TION ABOUT ALL YOUR PROGRAMS & F
ILES"
330 DATA"  * GENERATE TAPE WORK
FILE OF DIRECTORIES CONTAINED ON
 ALL DISKETTES"
340 DATA"  * PRINT (OPTIONAL) 'E
XTRACT & LOAD DIRECTORIES' REPOR
T #1"
350 DATA"  * STORE DISKETTE NAME
S & USAGE IN TABLE"
360 DATA"  * SORT DISKETTE NAMES
 TABLE"
370 DATA"  * PRINT (OPTIONAL) 'D
ISKETTE NAMES' REPORT #2"
380 DATA"  * SORT DISKETTE USAGE
 TABLE"
390 DATA"  * PRINT (OPTIONAL) 'D
ISKETTE USAGE' REPORT #3"
400 DATA"  * TRANSFER TAPE WORK
FILE TO DISK WORK FILE ON DRIVE
0"
410 DATA"  * READ DISK WORK FILE
 AND PRINT (OPTIONAL) DISKETTE J
ACKET"
420 DATA"      LABEL' DIRECTORY RE
PORT #4"
430 DATA"  * THIS PROGRAM PASSES
 THE DISK WORK FILE ON DRIVE 0 T
```

```
O PROG DIR2 AND/OR DIR3"
440 DATA"   * THIS PROGRAM LINKS
AND CALLS UP PROGRAM DIR2 OR DIR
3 ON DRIVE Ø"
450 DATA" "
460 DATA"Program DIR2 explanatio
n:"
470 DATA"   * THIS PROGRAM IS RUN
 AFTER PROGRAM DIR1"
480 DATA"   * READ DISK WORK FILE
 GENERATED BY PROGRAM DIR1 FROM
DRIVE Ø"
490 DATA"   * PRINT 'DISKETTE GUM
 LABELS' REPORT #5"
500 DATA"   * THIS PROGRAM LINKS
AND CALLS UP PROGRAM DIR3 ON DRI
VE Ø"
510 DATA" "
520 DATA"Program DIR3 explanatio
n:"
530 DATA"   * THIS PROGRAM IS RUN
 AFTER DIR1 OR DIR2"
540 DATA"   * READ DISK WORK FILE
 GENERATED BY PROGRAM DIR1"
550 DATA"   * SORT DISK WORK FILE
, IN RAM, TO PROGRAMS & FILES SE
QUENCE"
560 DATA"   * PRINT 'CONSOLIDATED
 DIRECTORIES' REPORT #6"
570 DATA"   * PRINT 'DISKS DIRECT
ORY SUMMARY' REPORT #7"
580 DATA" "
590 DATA"record layout of tape w
ork file & disk work file:"
600 DATA"   * RECORD SIZE = 42 BY
TES"
610 DATA"   * Ø1-Ø8 = Ø8 BYTE = N
AME OF PROGRAM OR DATA FILE"
620 DATA"   * Ø9-11 = Ø3 BYTE = E
XTENSION"
630 DATA"   * 12-12 = Ø1 BYTE = F
ILE FORMAT CODE"
640 DATA"                       *
 A = ASCII"
650 DATA"                       *
 B = BINARY"
660 DATA"   * 13-13 = Ø1 BYTE = F
ILE TYPE CODE"
670 DATA"                       *
 Ø = BASIC PROGRAM"
680 DATA"                       *
 1 = DATA FILE"
690 DATA"                       *
 2 = MACHINE LANGUAGE PROGRAM"
700 DATA"                       *
 3 = ASSEMBLY LANGUAGE PROGRAM"
710 DATA"   * 14-15 = Ø2 BYTE = N
UMBER OF GRANULES"
720 DATA"   * 16-21 = Ø6 BYTE = N
UMBER OF BYTES"
730 DATA"   * 22-41 = 2Ø BYTE = N
AME OF DISKETTE"
740 DATA"   * 42-42 = Ø1 BYTE = S
PECIAL CODE USED BY PROGRAM DIR1
"
750 DATA"                       *
 'X' IDENTIFIES EVERY OTHER DISK
ETTE DURING DIR1 AND DIR2"
760 DATA" "
770 DATA"Printer brands:"
780 DATA"   * ALL PROGRAMS PRINT
ON ANY PRINTER, HOWEVER, REPORT"
790 DATA"      TITLES ARE DESIGNED
 & CODED TO PRINT ENLARGED LETTE
RS USING"
800 DATA"      EPSON CODES.  TO MO
DIFY FOR OTHER PRINTERS, LOOK AT
 LINES:"
810 DATA"         * DIR1: Ø82Ø Ø83Ø
 268Ø 379Ø"
820 DATA"         * DIR2: Ø51Ø Ø96Ø
 1Ø2Ø 1Ø3Ø 1Ø4Ø 133Ø"
830 DATA"         * DIR3: Ø49Ø Ø74Ø"
840 DATA" "
850 DATA"High speed poke:"
860 DATA"   * THE FOLLOWING PROGR
AMS USE THE HIGH SPEED POKE FOR"
870 DATA"      FAST SORTING.  TO T
AKE OUT THE POKE, LOOK AT LINES:
"
880 DATA"         * DIR3: Ø82Ø"
890 DATA"END"
900 VERIFYON:CLS2:PRINT@1," DISK
ETTE DIRECTORIES HANDLER ";:PRIN
T@40," DOCUMENTATION ";
910 PRINT@129," ready THE PRINTE
R ";:PRINT@193," press ENTER WHE
N READY ";
920 I$=INKEY$:IFI$=""GOTO920
930 PRINT@289," PRINTING........
.. ";
940 PRINT@384,"";
950 READA$:IFA$<>"END"THENPRINT#
-2,A$:GOTO950
960 END
```

Designing Your Own Adventure

By George Firedrake and Art Canfil
Rainbow Contributing Editors

If you have never played a role playing game and want to begin playing, try a play-by-mail (PBM) game. Flying Buffalo Inc. created the play-by-mail industry. Anyone can learn to play these games. No previous gaming experience is required.

Begin by getting the rules for the game you play from Flying Buffalo Inc., Dept. GMA, P.O. Box 1467, Scottsdale, AZ 85252-1467. Below are names of PBM games and the prices for the rules.

— STARWEB	$2.00
— HEROIC FANTASY	1.00
— BATTLE PLAN	0.50
— NUCLEAR DESTRUCTION	0.25
— GALACTIC CONFLICT	1.00
— STARLORD	1.00
— BOARD OF DIRECTORS	0.25
— FEUDAL LORDS	1.00

Last time we suggested you sign up for *HEROIC FANTASY* and make a move every two weeks or once a month. First get the rules, then design a party of Adventurers and send them in as described in the rules, of course.

Your characters can be human or otherwise. Each character is a fighter or magic-user, but not both. The strength (STR) of a character is used to attack other characters or monsters, to defend oneself and others, cast

---

(George Firedrake, a.k.a. Bob Albrecht, is one of the most prolific authors in the microcomputer world today. A specialist in writing for beginners, he is author of numerous books including TRS-80 Color BASIC. *Art Canfil enjoys designing games and writing. He is co-author of* Taipan: A Game In Context.)

magic spells, and numerous other things. The constitution (CON) of a character determines the amount of damage a character can withstand and continue living. Each character type has a price (COST). Here are all possible character types.

CODE	KINDRED	CLASS	STR	CON	COST
F	Fairy	Fighter	1	1	1
F	Fairy	Magic-user	1	1	2
G	Gremlin	Fighter	3	4	3
L	Leprechaun	Magic-user	3	4	4
H	Hobbit	Fighter	5	15	5
H	Hobbit	Magic-user	4	15	7
K	Goblin	Fighter	7	20	6
P	Human	Fighter	15	30	9
P	Human	Magic-user	10	30	11
E	Elf	Fighter	25	25	15
E	Elf	Magic-user	20	25	18
D	Dwarf	Fighter	30	40	23
D	Dwarf	Magic-user	30	40	36
O	Ogre	Fighter	35	40	29
O	Ogre	Magic-user	35	40	46
T	Troll	Fighter	50	50	57
X	Giant	Fighter	60	60	72

For any character, you may choose the name and whether the character is male or female.

You assemble a party of Adventurers by "buying" up to 15 characters. You have 100 points to spend in acquiring characters.

You have 100 points to spend.

Let's try it. For our first group, how about a big guy and 14 tiny helpers? Our group consists of a giant and 14 fairy magic-users.

QTY	KINDRED	CLASS	STR	CON	POINTS
1	Giant	Fighter	60	60	72
14	Fairy	Magic-user	14	14	28
		TOTALS	74	74	100

Or, instead of 14 fairies, let's try seven leprechauns.

QTY	KINDRED	CLASS	STR	CON	POINTS
1	Giant	Fighter	60	60	72
7	Leprechaun	Magic-user	21	28	28
		TOTALS	81	88	100

The second bunch is higher in both STR and CON than the first group.

When Frodo, et al. set forth to return the ring to Orodruin, his group included hobbits, humans, elves, and dwarves (plus Gollum, of course). Let's put together our own *Fellowship of the Ring.*

QTY	KINDRED	CLASS	STR	CON	POINTS
2	Hobbit	Fighter	10	30	10
2	Hobbit	Magic-user	8	30	14
1	Human	Fighter	15	30	9
1	Human	Magic-user	10	30	11
1	Dwarf	Fighter	30	40	23
1	Elf	Fighter	25	25	15
1	Elf	Magic-user	20	25	18
			118	210	100

This Adventuring party has much more total CON and STR than either previous group. Of course, we really don't know what is important until we send one of our groups into the labyrinth and find out what happens.

YOUR TURN. Design your own bunch of Adventurers. Remember, you have 100 points to spend and you can select, at the most, 15 Adventurers. Choose a name for each character and decide who is male and who is female.

CoCo Can Help Design A Group
The ratios of STR to COST and CON to COST might be useful indexes to help design a group of Adventurers. Here are some examples.

Fairy fighter:	STR/POINTS=1	CON/POINTS=1
Fairy magic-user:	STR/POINTS=.5	CON/POINTS=1
Goblin fighter:	STR/POINTS=1.17	CON/POINTS=3.67

Goblins are durable, compared to their cost, while fairies are fragile, relative to their cost. You can buy a lot of CON for your money by stocking up on goblins!

We have in mind several programs to help design Adventuring teams and play *HEROIC FANTASY.* First, we need a database of information about character types. Here it is:

```
32000 REM**HEROIC FANTASY GMA 21
32002 REM**CHARACTER TYPES
32004 REM**CODE$,KIN$,CLASS$,STR
```

```
,CON,PTS
32010 DATA F,FAIRY,F,1,1,1
32020 DATA F,FAIRY,M,1,1,2
32030 DATA G,GREMLIN,F,3,4,3
32040 DATA L,LEPRECHAUN,M,3,4,4
32050 DATA H,HOBBIT,F,5,15,5
32060 DATA H,HOBBIT,M,4,15,7
32070 DATA K,GOBLIN,F,7,20,6
32080 DATA P,HUMAN,F,15,30,9
32090 DATA P,HUMAN,M,10,30,11
32100 DATA E,ELF,F,25,25,15
32110 DATA E,ELF,M,20,25,18
32120 DATA D,DWARF,F,30,40,23
32130 DATA D,DWARF,M,30,40,36
32140 DATA O,OGRE,F,35,40,29
32150 DATA O,OGRE,M,35,40,46
32160 DATA T,TROLL,F,50,50,57
32170 DATA X,GIANT,F,60,60,72
32180 DATA Z,ENDFILE,Z,0,0,0
```

This is a small data file consisting of 18 records. Each record contains information about one character type. For instance:

32010 DATA F, FAIRY, F, 1, 1, 1

CODE   KINDRED   CLASS   STR   CON   POINTS

Line 32004 tells you the names of the variables that we will use to store information from a *DATA* statement.

32004 REM**CODE$, KIN$, CLASS$, STR, CON, PTS

32060 DATA H, HOBBIT, M, 4, 15, 7

The last record, called ENDFILE, with CODE$ = "Z", is not a character type. It is the End-of-File (EOF) record.

32180 DATA Z,ENDFILE, Z, 0, 0, 0

End-of-File record
(No more records in the file.)

We have written two programs that use the data file of *HEROIC FANTASY* character types.

The *SCAN CHARACTER TYPES* program begins at Line 1000. It lets you scan the entire file. To run it, type *RUN* or *RUN 1000.* It begins like this.

```
F FAIRY        F  1  1  1
TO DO AGAIN, PRESS SPACE BAR
```

SPACE BAR is in
reverse color.
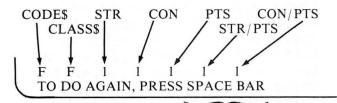
Press the space bar and you get the next record.

```
F FAIRY        F  1  1  1
F FAIRY        M  1  1  1
TO DO AGAIN, PRESS SPACE BAR
```

Keep pressing the space bar until you see 15 records on the screen. Press the space bar again to get the 16th record — the top record is "pushed off the top of the screen" and disappears.

Keep pressing the space bar until ENDFILE appears at the bottom of the screen. Press the space bar again and the CoCo starts over with the first record.

The *COMPUTE COST RATIOS* program begins at Line 2000. Type *RUN 2000* to run this program. First you see:

CODE$ STR   CON   PTS   CON/PTS
     CLASS$ STR/PTS

F   F   1   1   1   1   1
TO DO AGAIN, PRESS SPACE BAR
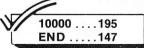
reverse color

This program works the same way as the *SCAN CHARACTER TYPES* program. Each time you press the space bar, you see another line of information near the bottom of the screen. If you see ENDFILE and press the space bar, the CoCo starts over at the top of the data file.

Here are both programs and the subroutines they use.

The listing:

```
10000 .... 195
END ..... 147
```

```
1 REM**HEROIC FANTASY GMA 21-1
1000 REM**SCAN CHARACTER TYPES
1010 CLS
1020 RESTORE        'START AT TOP
1030 GOSUB 11010 'READ RECORD
1040 GOSUB 12010 'SHOW RECORD
1050 GOSUB 10010 'TELL HOW AGAIN
1099 '
1100 REM**START OVER IF ENDFILE
1110 IF KIN$="ENDFILE" THEN 1020
 ELSE 1030
1199 '
2000 REM**COMPUTE COST RATIOS
2010 CLS
2020 RESTORE        'START AT TOP
2030 GOSUB 11010 'READ RECORD
2040 GOSUB 13010 'COST RATIOS
2050 GOSUB 14010 'SHOW RATIOS
2060 GOSUB 10010 'TELL HOW AGAIN
2099 '
2100 REM**START OVER IF ENDFILE
2110 IF KIN$="ENDFILE" THEN 2020
 ELSE 2030
2199 '
10000 REM**DO AGAIN SUBROUTINE
10010 PRINT @480, "TO DO AGAIN,
PRESS space bar";
10020 IF INKEY$="" THEN 10020
ELSE RETURN
10099 '
11000 REM**READ RECORD SUBR.
11010 READ CODE$,KIN$,CLASS$,STR
,CON,PTS
11020 RETURN
11099 '
12000 REM**SHOW RECORD SUBR.
12010 PRINT @480, CODE$ TAB(2)
KIN$ TAB(16) CLASS$ TAB(19) STR
TAB(23) CON TAB(27) PTS
12020 RETURN
12099 '
13000 REM**COST RATIOS SUBR.
13010 IF KIN$="ENDFILE" THEN
SC=0: CC=0: RETURN
13020 SC = STR/PTS
13030 SC = INT(100*SC+.5)/100
13040 CC = CON/PTS
13050 CC = INT(100*CC+.5)/100
13060 RETURN
13099 '
```

```
14000 REM**SHOW COST RATIOS SUBR
14010 PRINT @480, CODE$ TAB(2)
CLASS$ TAB(5) STR TAB(9) CON
TAB(13) PTS TAB(17) SC TAB(24)CC
14020 RETURN
14099 '
```

Of course, remember to add the data file (lines 32000 through 32180) to the programs.

— To run *SCAN CHARACTER TYPES*, type *RUN* or *RUN 1000* and press ENTER.
— To run *COMPUTE COST RATIOS*, type *RUN 2000* and press ENTER.

These are "bare bones" programs. Try your hand at improving them. Also think about other programs to help you design an Adventuring team and guide them as they explore the labyrinth.

Who Is A Character?

A character is any imaginary person or other creature created according to the rules of a game system. The characters in *HEROIC FANTASY* are quite simple. The characters in *Dungeons & Dragons* or *RuneQuest* are much more detailed and complex. Characters in *Adventurer's Handbook* are simplified versions of characters found in the very elegant *RuneQuest* system.

We need a way of recording a character's characteristics, abilities, knowledge, possessions, and anything else we want to remember. Below is a blank character record. You may copy it for your own use.

Character Name _____ Age _____ Sex _____ Race _____
Background _____

STR	_____		HIT PTS.	0	1	2	3	4	5	6	7	8	9	10
CON	_____			11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
SIZ	_____													
INT	_____	Idea _____	Armour _____
POW	_____	Luck _____	
DEX	_____	Dodge _____	POW PTS.	0	1	2	3	4	5	6	7	8	9	10
CHA	_____	Persuade _____		11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

SKILLS

		WEAPONS	DAMAGE	ATTACK	PARRY	BRK PTS.
CLIMB	_____	_____	_____	_____	_____	_____
FIRST AID	_____	_____	_____	_____	_____	_____
HIDE	_____	_____	_____	_____	_____	_____
JUMP	_____	_____	_____	_____	_____	_____
LISTEN	_____	_____	_____	_____	_____	_____
MOVE QUIETLY	_____	_____	_____	_____	_____	_____
SPOT HIDDEN	_____	_____	_____	_____	_____	_____
SWIM	_____					
THROW	_____	SPELL	%	SPELL	%	
	_____	_____	_____	_____	_____	
	_____	_____	_____	_____	_____	
	_____	_____	_____	_____	_____	

EQUIPMENT   MONEY
_____   _____
_____   _____
_____   _____
_____   _____

Last time we showed you the character record for Aloysious Anonymous, a very average character. Now meet Rokana.

Rokana is a beginning magic-user. She has learned three magic spells called HEALING, PROTECTION, and LIGHT. However, she has not yet mastered these spells. She has a 35 percent chance of successfully casting a HEALING spell and a 25 percent chance with PROTECTION or LIGHT.

Next time we will take Aloysious, Rokana, and perhaps some other characters to a county fair where they can have a wonderful time exercising their skills. In the meantime, we suggest you do some homework. Dig out the following back issues of THE RAINBOW and read "GameMaster's Apprentice."

August 1983 — pages 74-78
October 1983 — pages 170-174
November 1983 — pages 140, 144, 146, 148

Do any of you want us to run a small play-by-mail game? In this game, you would run one character like Aloysious or Rokana. You take your character to a county fair. Today they are called "Renaissance Faires," but in the world of Aloysious and Rokana they were contemporary fairs.

No previous experience is needed to play our play-by-mail game. Your only costs will be a copy of *Adventurer's Handbook* and some self-addressed, stamped envelopes. If you want to play, send a self-addressed, stamped envelope to DragonFun, P.O. Box 310, Menlo Park, CA 94026.

ROLE PLAYING GAMES

Millions of people play fantasy role playing games. A role playing game is a game in which one or more players create and play characters (adventurers) who live their imaginary lives in a specially made game world. The game world is created, managed, and operated by a GameMaster (GM), referee, or dungeon master (DM).

Most people who play role playing games use a formal rule system. Some of the best known are shown below.

Champions. Hero Games, 92A 21st Avenue, San Mateo, CA 94402.

Dungeons & Dragons (D&D). TSR, P.O. Box 756, Lake Geneva, WI 53147.

RuneQuest (RQ). Chaosium, P.O. Box 6302, Albany, CA 94706.

Traveller. Game Designer's Workshop, P.O. Box 1646, Bloomington, IL 61701.

Tunnels & Trolls (T&T). Blade, P.O. Box 1210, Scottsdale, AZ 85252.

Beginners beware! The rule books are formidable. If you are a beginner, we suggest you start with one of the following books, both from Reston Publishing Company, 11480 Sunset Hills Road, Reston, VA 22090.

Adventurer's Handbook: A Guide to Role Playing Games by Bob Albrecht & Greg Stafford.

Through Dungeons Deep by Robert Plamondon.

In "GameMaster's Apprentice," we include how-to-play information for all beginners.

Copyright© 1984 by DragonQuest, P.O. Box 310, Menlo Park, CA 94026.

One-Liner Contest Winner . . .
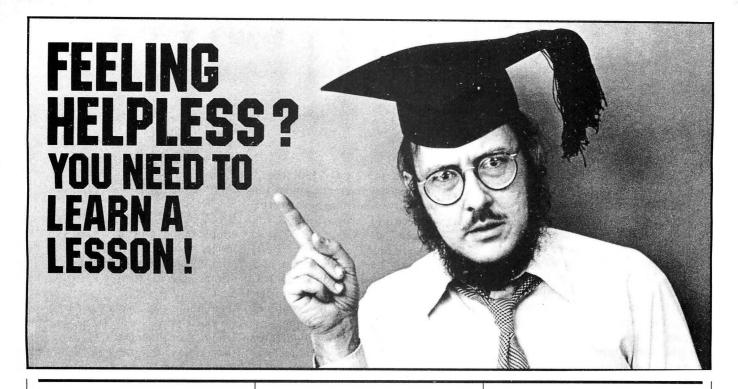
This shows the moon with many stars in the background. Let it run for a while (a half hour or so); it gets better with age!

Mark Perry
Rutherfordton, NC

The listing:

```
Ø PMODE4:PCLS:SCREEN1,1:CIRCLE(1
60,50),45,,.8:PAINT(160,50),1,1:
FORX=1TO99999999:PSET(RND(255),R
ND(192)):PRESET(RND(255),RND(192
)):NEXT'LET RUN FOR HALF A HOUR
OR SO.
```

(For this winning one-liner contest entry, the author has been sent copies of both *The Rainbow Book Of Adventure* and its companion *Rainbow Adventure Tape*.)

PIPELINE

SOME INTERESTING THINGS are being done in advanced information storage technology in Japan by Nippon Columbia, parent of Denon America, Inc. Nippon Columbia, credited with developing digital audio recording in 1972, has applied the technology of the Compact Disc to a very high capacity read-only memory storage system for personal computers. Dubbed the CD-ROM, the new disc has the same compact dimensions (4¾ inches) and uses the same type of laser pickup "player" mechanism as the audio Compact Disc.

The primary advantage of the CD-ROM is its 550-megabyte storage capacity on one side — the equivalent memory potential of 500 to 1,000 conventional 5¼-inch floppy disks. Additionally, the optical CD-ROM and its drive mechanism is free from magnetic mishaps and head/drive problems that plague the more conventional magnetic media and it has a built-in error detection and correction system.

For more information about the CD-ROM system, contact Robert Heiblim, Denon America, Inc., 27 Law Drive, Fairfield, NJ 07006; (201) 575-7810.

* * *

THE PRINTED WORD. Several recently released books are now available that Color Computer users might find interesting. First in the lineup are *The Computer Blue Book* and the *Orion Buyer's Guide to Computers*, published by Orion Research, 1315 Main Avenue, Suite 230, Durango, CO 81301. Both books are industry reference guides for both the buyer and seller in the marketplace and are available for $49.50 apiece.

Next we come to *Digital Deli*, authored by "the Lunch Group and Guests" and edited by Steve Ditlea. Billed as "the computer book with the human touch," this book combines contributions from 140 authors and artists into a 400-page *potpourri* of computer lore and culture and technological trivia. *Digital Deli* is available for $12.95 from Workman, 1 West 39 Street, New York, NY 10018.

If you're interested in success stories, there is *The Computer Entrepreneurs*, a look at the biggest names in the computer industry and how they made it to the top. Written by Robert Levering, Michael Katz and Milton Moskowitz, *The Computer Entrepreneurs* is an intimate portrait of these new cultural heroes. It is currently available for $19.95 from NAL Books, 1633 Broadway, New York, NY 10019.

Finally, we come to a two-volume reference set entitled *Abbreviations: The Comprehensive Dictionary of Abbreviations and Letter Symbols for the Computer Era* (it probably took two volumes just to get the name across the cover). The title pretty well says it all except that the volumes are arranged separately: one contains words to abbreviations and the other contains abbreviations to words. *Abbreviations* was compiled by C. Edward Wall and is currently available for $98 through Pierian Press, P.O. Box 1808, Ann Arbor, MI 48106.
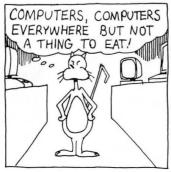
* * *
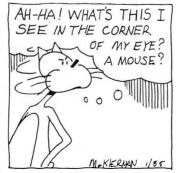
BACK ON THE LINE. Glancing at a couple of the information networks this month, CompuServe Incorporated has announced that they have entered into a joint agreement with VideoFinancial Services of Dallas, Texas, to expand their home and small business market for video banking by personal computer. Under the agreement, all CompuServe subscribers will be able to conduct transactions electronically with any bank in the country linked through VideoFinancial's computer gateway facilities. CompuServe will be the first nationwide computer network to offer this service, which is expected to be active on the system by March of 1985. For more information, contact Carl Byoir & Associates, Inc., 401 North Michigan Avenue, Chicago, IL 60611; (312) 527-5100.

The Business Computer Network has recently announced that they have added Knight-Ridder's VU/TEXT to their system. This means that their subscribers can now get the full text of the *Washington Post*, the *Philadelphia Enquirer* and the *Miami Herald*, with plans to have the *Boston Globe* online by the end of the year and the *Chicago Tribune*, *Detroit Free Press* and the Associated Press's *Datastream* soon after that. For more information, contact Business Computer Network, Inc., 1000 College View Drive, Riverton, WY 82501; or call Fred Rackmil at Howard J. Rubenstein Associates, (212) 489-6900.

CoCo Cat

Panel 1: WHAT... ANOTHER NEW STORE IN TOWN?

Panel 2: COMPUTERS, COMPUTERS EVERYWHERE BUT NOT A THING TO EAT!

Panel 3: AH-HA! WHAT'S THIS I SEE IN THE CORNER OF MY EYE? A MOUSE?

Panel 4: WHY DO I ALWAYS GET IN TROUBLE FOR TRYING TO DO SOME PEOPLE A FAVOR?

McKIERNAN 1/85

The Hi-Res lowercase punctuation *INTERPLANETARY* Fruit Fly Baby

By Martin Kaste

I'm sure most of you ambitious amateur programmers are familiar with the profound message of KISS, "Keep It Simple, Stupid!" Yet some people never seem to learn and continue slaving over thousand-line programs for months, only to watch them die slowly before their eyes on the screen, wondering where they went wrong. To me, the beauty of programming is that the most successful programs are usually the shortest, supported by a good idea, written in a few hours time and gradually perfected over a period of weeks.

This is true in the case of *The Interplanetary Fruit Fly*. It's short, simple and demonstrates a nifty little technique for displaying pseudo high-resolution graphics on the text screen: "lowercase punctuation."

You're probably asking, "A which?!"

(Martin Kaste is an American high school student living in Brazil. He develops programs for local home computer owners on his Color Computer.)

Let me explain. "Lowercase punctuation" is what I call all the periods, commas, question marks, brackets, etc., which your trusty old CoCo uses, with a special twist. Using *POKEs*, we can display these punctuation marks and other characters the same way the computer displays lowercase letters: the white symbol on a black rectangular background. With these characters, using a little ingenuity, we can create just about any figure we please.

These symbols can't be *PRINTed*, but, as I said before, the *POKEs* come to our rescue. I know some of you beginning programmers may shy away a little from *POKEs*. But don't worry, these *POKEs* are all addressed to the video RAM part of the computer, and can't interfere with its normal functioning, so bear with me.

The *POKE* addresses we are going to use range from 1024 to 1535, one location for every one of the 512 characters on the text screen. Except for the symbols we're interested in, most of the 256 characters available with *POKE* can be *PRINTed*, some only with the use of *CHR$* codes. To save

time, I have compiled a convenient list of the symbols we want and the value of each:

0 = @	38 = &	51 = 3
1-26 = alphabet	39 = '	52 = 4
27 = [	40 = (	53 = 5
28 = \	41 =)	54 = 6
29 =]	42 = *	55 = 7
30 = ↑	43 = +	56 = 8
31 = ←	44 = ,	57 = 9
32 = black rectangle	45 = -	58 = :
33 = !	46 = .	59 = ;
34 = "	47 = /	60 = <
35 = #	48 = 0	61 = =
36 = $	49 = 1	62 = >
37 = %	50 = 2	63 = ?

Now, let's get to the point. Type in the program listing, but be careful! There is one of those useful but occasionally dastardly speed up *POKEs* in Line 5. Before you run the program, save it on tape. If you want to run it before that, delete Line 5 and add it again when you're ready to save.

After the opening title, a green stripe appears at the top of the otherwise black screen displaying the current score, high score and number of shots remaining,

respectively. The game starts immediately, but can be stopped using the pause feature on the CoCo.

The Interplanetary Fruit Fly has already started to make irritating kamikaze dives at you, and all you have to defend yourself with are fifty shots of space age fruit fly repellent! You are controlling a cannon loaded with the stuff at the bottom of the screen, which you move with the left and right arrow keys. With the help of two *PEEKs,* your cannon has continuous smooth movement, meaning that it keeps traveling until you release the key. The fire button, which is the up-arrow key, also has this feature.

The action in the game is not difficult to explain. The movements of the cannon and the projectiles it fires are simple: *POKE* the figure into the new location, cover up its trail with black *POKE 32s.* The cannon travels at intervals of two spaces at a time, the projectile zips upward on a *FOR/NEXT* loop, jumping 32 spaces at a time for vertical movement. The Fly is a little more complicated, but not much. It flies down much the same way the projectile goes up, only the 32 spaces added each time are varied by a *SIN(X)* function, (Line 17). The result is a fly that weaves around and appears to dodge shots. As you probably guessed the flapping of its wings is an illusion caused simply by switching the parentheses back and forth, open and closed, in each step.
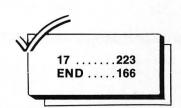
The player is awarded 500 points when he or she "wings" the Fly, 1,000 when the player manages to kill it, blasting the center of its body with the deadly repellent, and a new one takes its place almost instantly. The game ends if and when the Fly touches your cannon or when you run out of shots, whichever comes first. I suggest you ration your shots, because they go quickly, so try to shoot only when the Fly is in range. Another tip: try not to let yourself be fenced into the corners; they're perfect for the Fly to nail you.

One last note for those of you with Extended Color BASIC. You may want to liven up the game a bit by substituting the *SOUND* commands with faster, more exotic *PLAYs.* I use *PLAY"T 4505D04E03A#02DD01AADA"* in Line 10, *PLAY"T50D"* in Line 19, *PLAY"T50;01ADDE#D#AACA"* in Line 24, and *PLAY"T250;01A A05D03CCCCC"* in Line 28. You can think up something much better, I'm sure. Also, feel free to change any characters that I used in the game. The program layout is simple enough to allow you to make all kinds of modifications.

Enjoy and happy spritzing!

The listing:

```
0  '***THE*****INTERPLANETARY***
1  '******FRUIT****FLY**********
2  'BY***MARTIN*OLAF*KASTE*******
3  '**********MAY*1984***********
4  '*****************************
5  POKE65495,0
6  CLS0
7  FORA=1186TO1213
8  FORC=C TO C+5:POKEA,RND(26):NE
XTC
9  PRINT@162,MID$("the"+CHR$(128)
+"interplanetary"+CHR$(128)+"fru
it"+CHR$(128)+"fly",1,A-1186);:S
OUNDC,1:NEXTA
10 PRINT@189,"y";:FORA=1TO4:SOUN
D50,1:SOUND75,3:SOUND150,3:SOUND
2,1:SOUND3,1:PRINT@200,"by"+CHR$
(128)+"martin"+CHR$(128)+"kaste"
;:POKE1480,40:POKE1481,3:POKE148
2,41:POKE1483,32:PRINT@462,"smar
tsoft";:NEXT
11 FORN=1056TO1535:POKEN,32:NEXT
N
12 PRINT@0," "
13 PRINT@1,"0000";:PRINT@22,"SHO
TS:";:PRINT@29,"50";:IFHS=0THEN1
4ELSEPRINT@12,HS;
14 SH=50:XY=40:YX=41
15 A=1516
16 Z=1056+RND(32):IFSH=<0THEN24
17 POKEZ,32:POKEZ+1,32:POKEZ+2,3
2:IFZ>1503THEN16ELSEZ=Z+SIN(M)*R
ND(3):Z=Z+32:M=M+1:POKEZ,XY:POKE
Z+1,43:POKEZ+2,YX:CC=XY:XY=YX:YX
=CC
18 IFZ<A+3ANDZ>A-2THEN24
19 IFPEEK(341)=247THENFORC=A-31T
O1056STEP-32:POKEC,42:POKEC+32,3
2:NEXTC:POKEC+32,32:SH=SH-1:PRIN
T@28,SH;:IFPEEK(Z+1)=32THENS=S+1
000:GOSUB28:PRINT@1,S;:POKEZ,32:
POKEZ+1,32:POKEZ+2,32:GOTO16:ELS
EIFPEEK(Z)=32ORPEEK(Z+2)=32THENS
OUND50,1:S=S+500:PRINT@1,S;
20 IFSH=0THEN24
21 IF PEEK(343)=247THENB=-2:IFA=
1504THENB=0
22 IF PEEK(344)=247THENB=2:IFA=1
532THENB=0
23 POKEA,32:POKEA+1,32:POKEA+2,3
2:A=A+B:B=0:POKEA,47:POKEA+1,42:
POKEA+2,28:GOTO17
24 SOUND1,2:SOUND13,1:SOUND1,3:S
OUND2,4:SOUND1,2:FORA=1TO300:NEX
T
25 PRINT@33,"TO PLAY AGAIN, PRES
S THE A KEY";:A$=INKEY$:IFA$<>"A
"THEN25
26 IF HS<=S THEN HS=S
27 S=0:GOTO11
28 POKEZ,62:POKEZ+2,60:POKEZ-31,
22:POKEZ+33,1:SOUND50,1:SOUND150
,2:SOUND123,1:SOUND140,3:POKEZ,3
2:POKEZ+1,32:POKEZ+2,32:POKEZ-31
,32:POKEZ+33,32:RETURN
```

GRA

Graphics Quickies

The Line Box
By Kevin Oberberger

```
10 PMODE3,1:PCLS1:SCREEN1,1
20 CIRCLE(128,95),85,4:CIRCLE(12
8,95),60,4:PAINT(128,30),4,4
30 LINE(165,50)-(70,115),PSET:LI
NE(185,70)-(85,137),PSET:PAINT(1
00,95),4,4
40 CIRCLE(128,95),36,2,1,.26,.35
:DRAW"BM116,129C2M-2,+8M-10,+7BM
86,105M-6,-5"
50 CIRCLE(53,82),32,2,1,.1,.24:D
RAW"BM51,111F4G2H4L7F2L7G4H2E6L7
G2H2E4R2BL2M-4,-1H2L2U1E2M+10,+3
M+20,+1BL2M+14,-25M+6,-8M+2,-2M+
4,-2"
60 CIRCLE(88,57),18,2,1,.3,.7:CI
RCLE(108,54),18,2,1,.85,.2:CIRCL
E(102,30),20,2,1,.5,.8:DRAW"BM82
,40E2U7":CIRCLE(102,30),20,2,1,.
87,.1
70 CIRCLE(113,25),20,2,1,.65,.85
:DRAW"BM118,12E7R4D6M-8,+8":CIRC
LE(140,88),38,2,1,.7,.83:LINE(15
9,54)-(86,105),PSET:PAINT(90,80)
,1,2:PAINT(90,60),1,2
80 DRAW"BM173,81E7F2M+22,-8R2D2M
-10,+4G3BE2M+6,-1M+6,+1F2D1L9G2M
+10,+4G2M-10,-4L9G5L2H2U3L4H2":P
AINT(180,78),1,2
90 CIRCLE(94,29),2,2,2:CIRCLE(10
4,29),2,2,2:DRAW"BM96,56F2E2U2H2
G2D2":PAINT(98,55),2,2:DRAW"BM96
,66F2E3BU24G3H2BU22BL2G4BR18BU1H
3"
100 GOTO 100
```

PHIC

Nova
By Michael Rosenberg and Tobin Wonn

```
10 '      **** N O V A ****
20 '   BY MICHAEL ROSENBERG
30 '           AND
40 '        TOBIN WONN
50 '      MAY 10, 1984
60 PCLEAR8
70 PMODE 4,1:PCLS:SCREEN1,1
80 FORJ=0TO255STEP2
90 LINE(J,191)-(128,96),PSET:NEX
T
100 FORJ=0TO255STEP2
110 LINE(J,0)-(128,96),PSET:NEXT
120 FORJ=0TO191STEP2
130 LINE(128,96)-(255,J),PSET:NE
XT
140 FORJ=0TO191STEP2
150 LINE(128,96)-(0,J),PSET:NEXT
160 PMODE4,5:PCLS:SCREEN1,1
170 FORJ=0TO255STEP4
180 LINE(J,191)-(128,96),PSET:NE
XT
190 FORJ=0TO255STEP4
200 LINE(J,0)-(128,96),PSET:NEXT
210 FORJ=0TO191STEP4
220 LINE(128,96)-(255,J),PSET:NE
XT
230 FORJ=0TO191STEP4
240 LINE(128,96)-(0,J),PSET:NEXT
250 FORX=1105STEP4
260 PMODE3,X:SCREEN1,1:FORY=1TO1
00:NEXTY,X
270 GOTO250
```

No Ghosts
By Balinda Fortman

```
1 POKE65495,0
2 PMODE1,1
3 PCLS3
4 SCREEN1,0:POKE65314,248
5 FORX=3TO7
6 FORY=2TO6
7 FORZ=0TO3
8 COLORZ
9 A=0:B=255:C=0:D=191
10 LINE(A,C)-(B,D),PSET,B
11 A=A+Y:B=B-Y:C=C+X:D=D-X
12 IFA<255ANDC<191THEN10
13 NEXTZ
14 NEXTY,X
15 RUN
```

An Optical Illusion
By Bryan Kerr

```
1 ' AN OPTICAL ILLUSION
2 ' BRYAN KERR
3 ' 1195 BRIARWOOD DR
4 ' JACKSON, MS   39211
5 ' 8/17/83
10 PMODE3,1:PCLS:SCREEN1.0
20 A$="C4D30M-30,-15U30NM+30,+15
M+30,-15M+30,+15NM-30,+15D30M-30
,+15U30"
30 DRAW"S3XA$;BM-80,-40XA$;BM+80
,-40XA$;BM+80,+40XA$;BD80XA$;BM-
80,+40XA$;BM-80,-40XA$;"
40 B$="C1BM-30,+15C1NU8D2NM+8,+4
```

```
U2BM+30,-15":C$="C1BM+30,+15NU8D
2NM-8,+4U2BM-30,-15":D$="C1BU29N
M-8,+4NM+8,+4BD29"
50 E$="C4BM-30,-15M-40,-20D10M+4
0,+20U10M+10,-5M-40,-20M-10,+5M+
40,+20BM+30,+15"
60 F$="C4BD31D40NM-8,-4M+8,-4U40
BM-8,+4BM-8,-4ND40BM+8,+4BU31"
70 G$="C4BD1M+30,-15M+40,-20NM-8
,-4D10M-40,+20U10M-10,-5NM+40,-2
0M+10,+5M-30,+15"
80 DRAW"BM128,96C4XA$;BU2XB$;XC$
;XD$;BD80XD$;BM-80,-40XC$;XD$;BU
80XC$;BM+80,-40XB$;XC$BM+80,+40X
B$;BD80BM+80,+40XD$;XB$;XE$;BU80
XB$;BD80BM-80,+40XG$;XE$;BM-80,-
40XG$;BU80XF$;XG$;BM+80,-40XF$;B
M+80,+40XE$;XF$BM-80,+40XE$;XF$;
BM-80,+40XG$;XE$;XF$;
90 FOR C=2TO3:FORT=1TO15:READX,Y
:PAINT(X,Y),C,4:NEXTT,C
100 FOR T=1TO7:READX,Y:PAINT(X,Y
),4,4:NEXTT
110 DATA 128,94,130,153,128,30,1
60,74,188,60,68,60,96,74,96,49,1
58,49,68,120,188,120,116,100,140
,100,105,140,152,140
120 DATA 135,95,130,160,130,37,1
90,75,190,135,70,135,70,75,130,8
5,130,135,118,105,95,56,165,80,1
58,145,190,95,70,95
130 DATA 103,65,151,65,158,95,98
,95,103,125,158,125,128,0
140 GOTO140
```

The Hole
By Gregg A. Ford

```
1 'CIRCLE2
2 'GREGG FORD
3 '95 ELDRED BEDFORD,OHIO 44146
10 PMODE 4,1:PCLS:SCREEN 1,1:FOR
 X=10 TO 170 STEP 2
20 CIRCLE(128,96),X,7
30 NEXT
40 CLS:FOR X=10 TO 170 STEP 5
50 CIRCLE(128,96),X,7
60 NEXT
70 FOR X=10 TO 170 STEP 10
80 CIRCLE(128,96),X,2
90 NEXT
100 FOR X=0 TO 170 STEP 4
110 CIRCLE(128,96),X,3
120 NEXT
130 FOR X=0 TO 170 STEP 15
140 CIRCLE(128,96),X,0
150 NEXT
```

```
160 FOR X=10 TO 170 STEP 9
170 CIRCLE(128,96),X,4
180 NEXT
190 FOR X=10 TO 170 STEP 3
200 CIRCLE(128,96),X,5
210 NEXT
220 FOR X=10 TO 170 STEP 1
230 CIRCLE(128,96),X,6
240 NEXT
250 FOR X=10 TO 170 STEP 2
260 CIRCLE(128,96),X,7,2
270 NEXT
280 FOR X=10 TO 170 STEP 3
290 CIRCLE(128,96),X,8,2
300 NEXT
310 FOR X=0 TO 170 STEP 5
320 CIRCLE(128,96),X,0,2
330 NEXT
340 FOR X= 10 TO 170 STEP 7
350 CIRCLE(128,96),X,1,2
360 NEXT
370 FOR X=0 TO 170 STEP 1
380 CIRCLE(128,96),X,6
390 NEXT X
420 CLS:PRINT@232,"THAT'S ALL FO
LKS!"
430 GOTO 430
```

The Impossible Cube
By Stephen Lai

```
5 '     IMPOSSIBLE CUBE
10 PMODE4,1:PCLS:SCREEN1,0
20 DRAW"S44BM34,3NFR12NM-1,2F5NM
-2,1D12NH2L12NEH5NU12URNM+2,1ER3
D2L2F2NU8M+1,2ENR8U3NR3U2NR3U3NR
3HNELH3ND7M-1,-2M+3,1NF3R7F4D9UH
3U4NR3U2NR2U2HLD3NL3D5FNRGF3"
30 FORF=1TO8:READA,B:PAINT(A,B):
NEXT:DATA0,0,100,30,60,70,140,90
,190,90,170,50,140,160,76,140
40 GOTO40
```

Pascal's Triangle
By Stephen Lai

```
5 '     PASCAL'S TRIANGLE
10 DIM P(256)
20 PMODE 4,1:PCLS:SCREEN 1,1
30 P(128)=1
40 FOR S=1 TO 127
50 FOR F=128-S TO 128+S STEP 2
60 P(F)=P(F-1)+P(F+1)-SGN(P(F-1)
+P(F+1))*2
70 PSET(F,S,P(F)+5)
80 NEXT F,S
90 GOTO 90
```

Ever become exasperated with the acrobatics required by the SHIFT @ method of stopping a fast-scrolling LIST? Or have you sometimes wanted to format the margins and line length of your LLISTINGS? Now you can easily accomplish either of these things by employing one of the useful processes described here.

SCROLL CONTROL AND LLIST MASTERR

By H. Allen Curtis

This article concerns two programs which produce new varieties of CoCo's *LIST* and *LLIST* commands. The programs apply regardless of the size of RAM or the type of BASIC in your CoCo.

Have you ever become exasperated trying to use SHIFT @ to stop the listing process at an appropriate place? The program of Listing 1 was written to avert such exasperation. This program offers you the option of two different varieties of *LIST* commands. Henceforth, the program of Listing 1 will be referred to as *Lister*.

Under the first option, *Lister* will produce a slow *LIST* command. With that *LIST* command, the scrolling of the BASIC lines will take place slowly. The slowed scrolling gives you the needed SHIFT @ control in the strategic stopping of the listing process. This *LIST* variety can operate at any one of five different speeds. You may choose the speed you find to be most convenient.

The second *LIST* variety operates at the normal speed, but without scrolling.

(H. Allen Curtis resides in Williamsburg, Va. He is interested in 17th and 18th century history and enjoys biking through the colonial capital. He balances past and present with his computer work.)

With this variety, when the screen is filled to a point where scrolling would ordinarily occur, listing is automatically stopped. You then must press ENTER for the screen to clear and the listing to resume at the top of the screen. Rather than stopping the listing process with SHIFT @, you must press ENTER to start it.

"... you may modify LIST *and* LLIST *to meet your particular needs. Disassemblies of the machine language subroutines associated with* Lister *and* LLister *should prove helpful in making modifications."*

The key to producing the two *LIST* varieties is the employment of the "hook" at RAM addresses 422 through 424. This hook connects the ROM routine for the *LIST* command with RAM. This hook is not used by Color BASIC, Extended Color BASIC or Disk BASIC. Therefore, it is available to use in the

alteration and improvement of the *LIST* command.

In the slow *LIST* variety, the hook is made to connect the ROM's *LIST* routine with a machine language subroutine residing at the 61 highest RAM locations of your CoCo. In the no-scroll variety of *LIST*, the hook connects the *LIST* routine and a 43-byte machine language subroutine stored in high RAM.

The values in the *DATA* statements of lines 180 and 190 of *Lister* are the 16 bytes comprising the subroutine associated with the slow *LIST*. Likewise, the values of the remaining *DATA* statements, lines 200 through 260, are the bytes that comprise the subroutine associated with the no-scroll *LIST* command.

Lines 10 through 40 are employed to store the subroutine of slow *LIST* in high RAM as well as setting the appropriate hook address. Lines 90 through 110 along with Line 130 serve a similar purpose for the no-scroll *LIST*. Line 170 completes the connection between the ROM's *LIST* and the selected variety. Line 120 checks the accuracy of your typing of the *DATA* statements.

Except for Line 5, the other lines of *Lister* are concerned with prompts to aid you in taking your *LIST* variety options. The *REM* of Line 5 informs

Listing 1:

```
5 REM TO RETURN TO NORMAL LIST
      POKE422,57
10 X=256*PEEK(116)+240:CLEAR100,
X-1
20 X=256*PEEK(116)+240
30 FORI=0TO15:READA:POKEX+I,A:NE
XT
40 POKE423,PEEK(116):POKE424,240
50 CLS:PRINT@134,"CHOOSE LIST VA
RIETY":PRINT@166,"BY PRESSING 1
OR 2
60 PRINT:PRINT"  1. SLOW":PRINT:
PRINT"  2. NO SCROLL
70 K$=INKEY$:ON VAL(K$) GOTO140,
90
80 GOTO70
90 X=256*PEEK(116)+213:CLEAR100,
X-1
100 X=256*PEEK(116)+213:FORI=0TO
15:READA:B=B+A:NEXT
110 FORI=0TO42:READA:B=B+A:POKEX
+I,A:NEXT
120 IFB<>6844THENCLS:PRINT"DATA
ERROR":STOP
130 POKE424,213:POKE422,126:END
140 CLS:PRINT@164,"FOR A FASTER
SLOW LIST":PRINT"         PRESS A NUM
BER KEY FROM":PRINT"            1 TH
ROUGH 4.":PRINT"        OTHERWISE, P
RESS ENTER.
150 K$=INKEY$:K=VAL(K$):IFK<5 AN
D K>0 THEN160ELSEIFK$=CHR$(13)TH
EN170ELSE150
160 K=256/2^K:POKE256*PEEK(116)+
249,INT(K+.5)
170 POKE422,126:END
180 DATA 52,16,134,254,145,111,3
9,6
190 DATA 142,0,0,189,167,211,53,
144
200 DATA 220,136,159,66,227,132
210 DATA 147,66,16,131,5,224,44
220 DATA 1,57,182,2,220,129,155
230 DATA 39,248,129,182,39,244
240 DATA 52,16,189,163,144,189
250 DATA 169,40,48,95,189,185
260 DATA 156,12,137,53,144
```

you how to return to the normal *LIST* from either variety of *LIST*.

The subroutines of both *LIST* varieties will occupy the same RAM area as RAINBOW Check Plus. Therefore, if you make use of RAINBOW Check Plus as an aid in the correct typing of *Lister*, do not run *Lister* until it has been completely and accurately typed.

After correctly typing *Lister*, save it on tape or disk, then *RUN* it. You will quickly receive the first prompt. To test

Lister initially, press the '1' key to select the slow *LIST* variety. Doing this immediately produces another prompt. The slow *LIST* will operate at one of five speeds. Pressing ENTER will make it operate at its slowest speed. Pressing 1, 2, 3 or 4 will, respectively, make it operate at two, four, eight or 16 times the speed of the slowest *LIST*.

Press ENTER to get the slowest *LIST*. Then type *LIST*, press ENTER, and test the use of SHIFT @ to stop the listing

process. Run *Lister* four more times. Each time select the slow *LIST* variety, but at a different speed. Determine what speed is most convenient for you by *LIST*ing *Lister* and stopping it by means of SHIFT @.

In another run of *Lister*, select the no-scroll *LIST* variety, then *LIST* the program. Note that the listing stopped after completing only BASIC Line 5. Listing Line 10 would have caused scrolling, so the process was stopped.

Listing 2:

```
10 CLS:X=256*PEEK(116)+194:CLEAR
100,X
20 X=256*PEEK(116)+195
30 FORZ=X TOX+60
40 READA:B=B+A:POKEZ,A:NEXT
50 IFB<>5998THENPRINT"DATA ERROR
":STOP
60 PRINT@231," ";:INPUT"MARGIN";
M
70 POKEX+60,M
80 INPUT"           LINE LENGTH";L
90 L=L+M:POKEX+43,L
100 IFPEEK(32768)<>69THENPOKEX+1
1,57
110 IFPEEK(186)=6THENPOKEX+12,13
0:POKEX+13,115ELSEPOKEX+12,203:P
OKEX+13,74
120 POKE360,PEEK(116):POKE361,19
5
130 DATA 52,124,246,2,220,193
140 DATA 155,39,5,53,124,57,0,0
150 DATA 48,140,42,129,13,39,34
160 DATA 230,132,38,16,230,1,52
170 DATA 6,231,132,134,32,189
180 DATA 162,133,90,38,248,53,6
190 DATA 92,193,80,39,4,231,132
200 DATA 32,215,189,162,133,134
210 DATA 13,111,132,32,206,0,0
```

Pressing ENTER permits the nearly full screen display of lines 10 through 80. By pressing ENTER again, you will be shown lines 90 through 140. Two more similar presses of ENTER will complete the program listing. As you have seen, the no-scroll *LIST* lets you examine BASIC lines at your own pace.

Setting Margins With *LLister*

Would you like to start a notebook of program listings? Or, would you like to have the ability to select the left margin size and the number of characters per line of any hard copy listing? If so, the program of Listing 2 should be of interest to you. The product of this program, called *LLister*, is a new variety of *LLIST* command that allows you to format your hard copy listings insofar as left and right margins are concerned.

Another hook, at addresses 359 through 361, is the key to the development of a new variety of *LLIST* command. The *DATA* statements (lines 130 through 210) of *LLister* comprise a 61-byte machine language subroutine that will be connected via the hook to the ROM's *LLIST* command routine. Lines 10 through 40 of *LLister* are concerned with storing the aforementioned subroutine at the highest 61 RAM locations. Line 50 provides a check on the accuracy of your *DATA* values typing.

Both Extended Color BASIC and Disk BASIC make use of the hook at addresses 359-361, but Color BASIC does not. *LLister* sets a hook address entry to the 61-byte machine language subroutine. For systems with Extended Color BASIC or Disk BASIC, *LLister* also sets the subroutine exit address, which is the replaced hook address entry to the appropriate BASIC ROM. Lines 100 through 120 are used to set up the hook address and the exit address, if needed.

The remaining lines of *LLister* provide prompts to enable you to specify the desired left margin size and the number of characters per line.

A high RAM conflict exists between *LLister* and RAINBOW Check Plus. When using the latter, do not run *LLister* until it has been typed correctly. If you are using a new *LIST*, type and ENTER *POKE422,57* to disable it before running *LLister*.

When you have finished the required typing, save *LLister* on tape or disk. Turn your computer off and turn on your printer and CoCo in the recommended sequence. *LOAD LLister* and *RUN* it.

A MARGIN prompt will soon appear

ADDRESS	HEX CODE	MNEMONIC	OPERAND	COMMENT
3FF0	34 10	PSHS	X	;Save X
3FF2	86 FE	LDA	#FE	;A is printer device number
3FF4	91 6F	CMPA	6F	;Is LLIST current command?
3FF6	27 06	BEQ	3FFE	;If so, go to exit
3FF8	8E 00 00	LDX	#0000	;X is delay constant (POKEd from BASIC for speeds 1 thru 4)
3FFB	BD A7 D3	JSR	A7D3	;Delay
3FFE	35 90	PULS	PC,X	;Restore X and exit

Table 1

ADDRESS	HEX CODE	MNEMONIC	OPERAND	COMMENT
3FD5	DC 88	LDD	88	;D is cursor position
3FD7	9F 42	STX	42	;Save line address pointer
3FD9	E3 84	ADDD	,X	;Add next pointer
3FDB	93 42	SUBD	42	;D is cursor position if current line were listed
3FDD	10 83 05 E0	CMPD	05E0	;Would cursor be on bottom screen line?
3FE1	2C 01	BGE	3FE4	;If so, branch
3FE3	39	RTS		;Exit
3FE4	B6 02 DC	LDA	02DC	;A is current command token
3FE7	81 9B	CMPA	#9B	;Is command LLIST?
3FE9	27 F8	BEQ	3FE3	;If so, go to exit
3FEB	81 B6	CMPA	#B6	;Is command EDIT?
3FED	27 F4	BEQ	3FE3	;If so, go to exit
3FEF	34 10	PSHS	X	;Save X
3FF1	BD A3 90	JSR	A390	;Wait for ENTER to be pressed
3FF4	BD A9 28	JSR	A928	;Clear screen
3FF7	30 5F	LEAX	-1,U	;X points to ASCII representation of current line number
3FF9	BD B9 9C	JSR	B99C	;Print line number
3FFC	0C 89	INC	89	;Bump cursor position
3FFE	35 90	PULS	PC,X	;Restore X and exit

Table 2

on the screen. The prompt requires you to type the margin size. A frequently used margin size is 10 characters long. Thus, type 10 and press ENTER.

The next prompt asks you to specify the line length. For an 80-column printer, 60 characters per line will yield left and right margins of 10 characters each. Therefore, type 60 and press ENTER.

Test the new *LLIST* command by using it to *LLIST LLister*. *RUN LLister* several times and try *LLISTing* with different margin and line length selections. If the selected margin size and line length total more than 80 characters, *LLister* will reduce the line length to 80 minus the margin size.

LLister as presented does not have any effect on the *PRINT#-2* command, but it can easily be made to do so. If you would like to employ *LLister* to select and use margins and line lengths on other matter as well as BASIC programs, make the following changes in *LLister*: in Line 130 replace values 246, 2 and 220 with 214, 111 and 18. In Line 140 replace the value 155 with 254. Finally, in Line 50 replace the checksum 5998 with 5972.

As *Lister* and *LLister* were written, the new varieties of *LIST* and *LLIST* cannot concurrently be a part of BASIC's vocabulary. The new *LIST* and *LLIST* commands can be made to belong to BASIC's repertoire by making the following changes in *Lister*: in lines 10, 20 and 40 replace the number 240 with 179. Similarly, in lines 90, 100 and 130 replace 213 with 152. Then save the changed *Lister* on tape or disk.

When adding both the new *LIST* and *LLIST* commands to BASIC's vocabulary, you must *LOAD* and *RUN LLister* before doing the same with the changed *Lister*.

Some of you may be interested in further modifying *LIST* and *LLIST* to meet your own particular needs. Disassemblies of the machine language subroutines associated with *Lister* and *LLister* should prove helpful to you in making such modifications. Disassemblies of the slow *LIST*, no-scroll *LIST* and *LLIST* subroutines are presented in Tables 1, 2 and 3.

In the tables all numbers and addresses have been expressed in hexadecimal, so there is no need to use the symbol '$' to indicate that a given number or address is in hexadecimal. The table addresses correspond to those used with a CoCo having a 16K byte RAM. For a 32K RAM mentally add 4000 hexadecimal to each of those addresses.

ADDRESS	HEX CODE	MNEMONIC	OPERAND	COMMENT
3FC3	34 7C	PSHS	UXYB,DP	;Save registers
3FC5	F6 02 DC	LDB	02DC	;B is command token
3FC8	C1 9B	CMPB	#9B	;Is it LLIST token?
3FCA	27 05	BEQ	3FD1	;If so, branch
3FCC	35 7C	PULS	UXYB,DP	;Restore registers
3FCE	7E 00 00	JMP	0000	;Exit set by POKEs
3FD1	30 8C 2A	LEAX	2A,PC	;X points to current character count
3FD4	81 0D	CMPA	#0D	;Is character carriage return?
3FD6	27 22	BEQ	3FFA	;If so, branch
3FD8	E6 84	LDB	,X	;B is current character count
3FDA	26 10	BNE	3FEC	;Branch if not start of new line
3FDC	E6 01	LDB	1,X	;B is margin size
3FDE	34 06	PSHS	B,A	;Save margin size and current character
3FE0	E7 84	STB	,X	;Set character count to margin size
3FE2	86 20	LDA	#20	;A is space
3FE4	BD A2 85	JSR	A285	;Line print space
3FE7	5A	DECB		;Have all margin spaces been printed?
3FE8	26 F8	BNE	3FE2	;If not, loop
3FEA	35 06	PULS	B,A	;Restore margin size and current character
3FEC	5C	INCB		;Bump B
3FED	C1 50	CMPB	#50	;Have all characters of current line been printed?
3FEF	27 04	BEQ	3FF5	;If so, branch
3FF1	E7 84	STB	,X	;Save current character count
3FF3	20 D7	BRA	3FCC	;Prepare to exit
3FF5	BD A2 85	JSR	A285	;Line print final character of line
3FF8	86 0D	LDA	#0D	;A is carriage return
3FFA	6F 84	CLR	,X	;Zero character count
3FFC	20 CE	BRA	3FCC	;Prepare to exit
3FFE	00			;Current character count
3FFF	00			;Margin size

Table 3

Like Sands Through The Hourglass...

By Mary Crooks

*H*ourglass is a short graphics routine for visual entertainment. It fills a time delay or visual gap in the on-screen proceedings. You can use it as a subroutine in programs you create that need such a time filler. If you add it to the "20 Odd-Ball Screen Rou-

(Mary Crooks, a housewife and mother of two, has been programming for about a year. She is currently helping to organize and run the computer center at her son's elementary school. She holds a degree in art and enjoys all things creative.)

tines" by Gregory Clark in THE RAINBOW, Jan. 1984, then you will have quite a collection from which to choose.

A Color Computer 16K Extended BASIC is required. Type *RUN* and the program writes the words "Please Wait" on the screen, then draws an hourglass. The sand in the hourglass falls for an interval of time that you can change to fit your needs. Lines 10-20 reserve memory for variables and determine the graphics mode. Lines 30-110 draw the cursive letters and assign them a variable name. The actual drawing of the words happens in Line 120 using *DRAW*

and concatenation (adding together) of the variables. The rest of the program constructs and colors the hourglass and creates the falling sand particles. The time delay is for ten seconds. You can change the timing interval in Line 230 by increasing or decreasing the 50 in the *FOR NEXT* loop.
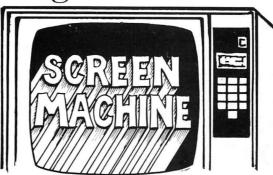
If you use *Hourglass* as a subroutine, include *CLEAR 350* at the start of your main program and delete it from the subroutine. Call the subroutine with a *GOSUB* from the place in your program that it is needed, then change Line 260 to read *RETURN*.

The listing:

```
10 CLEAR 350
20 PMODE4,1:PCLS:SCREEN 1,0
30 AP$="EU2UNHNUER2FDGL2HBD3BR4"
40 BL$="BRRE2U3HGD4FR"
50 BE$="RE2HGF2R"
60 BA$="RU2ERFDNFGLHBDBR4"
70 BS$="REU2F2GNHR2"
80 AW$="BU5FD3FENU4FEU3EBD5"
90 BI$="REU2BUNEBD3DRBR"
100 BT$="REU3NL2NR2NUBD3FR"
110 SS$="BR5"'SPACE
120 DRAW"S8BM20,20"+AP$+BL$+BE$+
BA$+BS$+BE$+SS$+AW$+BA$+BI$+BT$+
SS$
130 CIRCLE(126,60),30,,.25
140 CIRCLE(126,152),30,,.25
150 LINE(96,60)-(156,152),PSET
160 LINE(156,60)-(96,152),PSET
170 LINE(116,90)-(136,90),PSET
180 PAINT(118,91),1,1
190 PAINT(126,152),1,1
200 FOR Q=1TO 3
210 FOR T=1 TO 40
220 P=124+RND(4):Q=110+RND(26)
230 PSET(P,Q,1):FOR L=1TO 50:NEX
T
240 NEXT T:CIRCLE(126,152),15,,.
75,.50,1:PAINT(126,143),1,1
250 NEXT Q:CIRCLE(126,152),19,,.
75,.50,1:PAINT(126,140),1,1
260 FORX=1TO2000:NEXT
```

A Practical Alternative For Creating A Pupil Database

By Michael Plog Ph.D.
Rainbow Contributing Editor

Recently, I had a chance to talk with some school people about using a pupil database. These people were directors of a program operating in schools, and needed to supply information to funding agencies (state education agencies, Federal grant agencies, or even foundations and industries). The funding agencies request information about each student being served with the money provided. Thus, the pupil database is a set of information wherein each student represents one record. Specific data elements may include such things as birth date, grade level, sex, scores on tests, date of entering the program, length of service by program personnel, and so on.

A pupil database is ideally suited for a microcomputer. There is the need to store information, have access to that information for updating, and prepare a report to be delivered to the funding agency. In this article, we will consider

(Michael Plog received his Ph.D. degree from the University of Illinois. He has taught social studies in high school, worked in a central office of a school district, and currently is employed at the Illinois State Board of Education.)

some of the features of a pupil database, and some of the uses.

With many commercial databases on the market, there is no need for local programmers to try to write their own database. Local efforts will probably not be as flexible as commercial programs, and will have to go through a long period of debugging. The time required to prepare a complete database program is probably not worth the amount of money required to buy one of several excellent products. I have no idea how many database packages are available for the Color Computer, but I have seen some in operation. With the power available for the low price, it is silly for local people to try to write their own. Thus, it is suggested that school personnel buy a commercial product; one that fits their needs and is easily understood.

When designing the database, and determining what information to store, include more than the minimum required by the funding agency. Consider what will be needed for local school use, in addition to what the outside agency will need. Such things as parents' names, address, telephone number, grades in classes, etc., are of local concern, but generally not of concern

to funding agencies. Design the database carefully. Remember, it will be impossible to get anything out of the pupil database that has not been put in.

Getting information out of a database may prove more complex than most people think. To get local reports from a database, the commercial reporting formats may work fine. However, the report to the funding agency may be a different matter, and after all, what is the sense of storing information on a microcomputer if you cannot have the machine prepare the reports for you?

Most funding agencies now use computer printer paper for student reports. After all, they have to enter the information they receive (typically on a mainframe), and simply use what is convenient. The trouble is that most of these forms will not be appropriate for commercial database report formats. Such things as the name of the school district may have to be printed on each page. Line spacing may prove to be a difficulty. Summary calculations may be required. Judging from the student reports I have seen, commercial database report formats are inadequate for the job.

The best solution to having the microcomputer print the necessary

reports is to write a program to take the data from storage, format it in the required manner, and print it directly to the forms provided by the funding agency. If the records are stored in ASCII format, a BASIC program will solve this problem easily.

But there is the rub. Most program directors would find that hard to do. Few would have the time necessary to write such a program in BASIC. Even if program directors had the time, few have the background in programming. Generally, their training has not included a study of computer languages. Naturally, there will not be a major inclination to learn BASIC just for this purpose.

Hiring professional programmers is expensive. And while I have nothing against people selling their programming skills (I have done so myself), it must be recognized that many schools simply cannot afford the luxury of contracting with a programmer for what may well be a small program affecting only a few students. What is needed is inexpensive

> *"It seems an entirely workable idea to have students put classroom experiences to use on a real life problem."*

programmers. (I refuse to say cheap programmers, because that implies low quality.)

There is a solution which makes this difficulty minor. Why not use students in school taking computer classes? The majority of schools in the country are teaching classes in BASIC. Students in these courses are learning a great deal about programming, and even writing programs that get published in magazines such as THE RAINBOW. The instructors of these courses are right in the buildings with the projects needing assistance from microcomputers. And of course, the price is right.

It seems an entirely workable idea to approach the instructor of a computer course with the idea of having the students put their classroom experiences to use on a real life problem. This would not only help the local district (getting reports in on time), but would give the students a chance to show their creativity, and progress beyond the typical exercises and activities of the classroom. The task would provide an excellent opportunity for the more advanced students in the computer course, but

would also be appropriate as a project for the entire class.
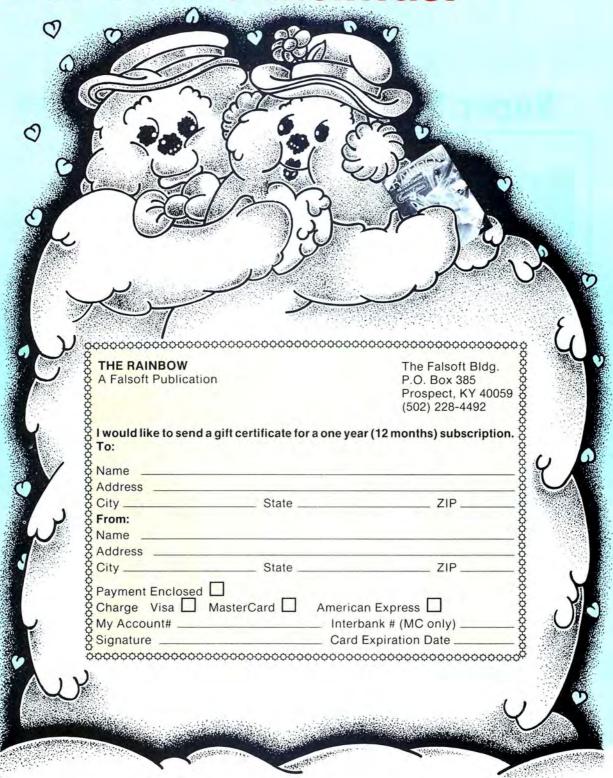
I realize that most readers of this article are not going to be program directors in need of microcomputer assistance. A more likely audience will be instructors of the computer classes. Well, go ahead — take the initiative. If you think your class could accomplish this type of activity, seek out those directors and coordinators of programs that might benefit from such a program. They will not be hard to find. Such folks are involved in things like special education, bilingual, chapter I, vocational programs, or in Federal programs funded directly to schools. Sometimes foundations and large industrial firms will fund special programs in schools.

The point of this suggestion is to advance and expand the use of microcomputers in schools. This is perhaps a non-traditional use of the microcomputer, and certainly a non-traditional use of microcomputer courses. But the cooperation that results cannot be anything but helpful — to students, teachers, program directors, and the school system itself.

Another thought about the database concept needs to be mentioned. The initial data load can be a problem, much more so than simply updating information already stored. Most school systems do not have enough secretarial help to keep up with the daily tasks that have to be done, much less take the time to load a host of student information into a microcomputer. What is needed is an inexpensive source of typing labor. Since we are using inexpensive programmers, why not continue the same idea here. Approach the typing teacher. I would imagine most teachers would appreciate the opportunity to have their students gain experience with a different keyboard, learn something about data entry, and get some practice on an activity that has immediate value.

So, it should be possible to use the microcomputer to further the cause of cooperation in schools, provide services that would not be available otherwise, and broaden the potential uses of modern technology in education. Not a bad deal. If anyone is currently engaged in such an activity, please let me know. If anyone starts something like this, I would appreciate knowing about it. My address is 829 Evergreen, Chatham, IL 62629. Until next month, take care and keep learning.

It'sno job to give the perfect gift for this Christmas.

Cooking With CoCo

Part VII

By Colin J. Stearman

Teaching CoCo how to clean up after its errors and own up to mistakes

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

Probably the most frustrating limitation of the Microsoft BASIC in CoCo is its lack of ability to trap errors. Even the best written programs generate errors and when they do, it's infuriating to have CoCo tell you how you messed up and then tell you with a condescending smirk that it's OK! It isn't OK, so we must do something about it.

Error Trapping

Most flavors of BASIC have a statement similar to *ON ERROR GOTO nnn* which tells the interpreter that if an error occurs jump to line 'nnn' and continue running. Then at line 'nnn' we can write some lines which handle the error and continue the running of the program.

Because *ON* is already a BASIC keyword I decided to simplify the syntax. So here is a description of the error trapping command and some associated variables.

ERRORS

The syntax for the error directing line is *ERRORS GOTO nnn*, where 'nnn' is an existing line number or zero. When such a line is encountered in your program it simply tells the interpreter that, should an error occur, go to line 'nnn'. This command will stay in effect until another such line is encountered saying go to a different line on an error. Except if 'nnn' is a zero, error trapping is canceled and errors cause BASIC to stop the program and report just as before (or nearly as before, as you will see).

If line 'nnn' does not exist, then a 'No such line number' error will occur if the statement is entered in the direct mode. However, if it is in a program, it will create an error itself, but the error will have nowhere to go, and the program will lock up. Pressing Reset is the only option left.

Because the line number follows a normal *GOTO* statement, the *RENUM*

command will handle it correctly.

When any error occurs all *FOR . . . NEXT* loops and subroutine return addresses are canceled, allowing the error handling routine to jump to anywhere in the program without a problem.

ECODE

This numeric variable returns the current error code number. If no error has yet been encountered, it will have the value -1, so if a NO SUCH FILE error was the most recent error, then doing a PRINT ECODE would print 26, the code number for that error. ECODE may be used just as any other numeric variable, but it may not be assigned a value by putting it on the left of an equal sign.

ELINE

This is also a numeric variable and all comments about ECODE apply equally to it. This returns the BASIC line number on which the most recent error occurred. If no error has yet occurred this variable will have the value of -1.

ENAME$

This is a string variable which contains the name of the most recent error. If no errors have yet occurred, ENAME$ is a zero length string. All normal string manipulation functions may use it, but it too must not appear on the left of an equal sign.

The error code numbers returned by ECODE and the associated error strings are:

ECODE	ENAME$
0	NEXT without FOR
1	Syntax
2	Return without GOSUB
3	Out of Data
4	Function Call
5	Overflow
6	Out of Memory
7	No such line #
8	Subscript
9	Redimensioned Array
10	Divide by 0
11	Illegal Direct Command
12	Type Mismatch
13	Out of String Space
14	String too long
15	String too complex
16	Can't Continue
17	File Data
18	Already Open
19	Device Number
20	Read/Write
21	File Mode
22	File Not Open
23	Read past End of File
24	Direct Command in File
25	Undefined Function
26	No such File
27	Record #
28	Disk Full
29	Out of Buffer Space
30	Write Protect
31	File Name
32	Directory
33	File Exists
34	Field Overflow
35	Set to Non-Fielded String
36	Verify
37	Access past End of File

If no error trapping is set, BASIC will return these fully spelled out error messages followed by the word ERROR, instead of the cryptic question mark and two letter code.

Due to memory space limitations, ENAME$ and fully spelled out error messages are not included in the patch to *DECB 1.1.*

SWAP

The final BASIC command to be added is SWAP. This has no connection with error trapping but is useful to have around. The syntax is: *SWAP var1,var2*

> *"When any error occurs all FOR . . . NEXT loops and subroutine return addresses are canceled, allowing the error handling routine to jump to anywhere in the program without a problem . . . If no error trapping is set, BASIC will return these fully spelled out error messages followed by the word ERROR, instead of the cryptic question mark and two letter code."*

where 'var1' and 'var2' are like variables. This means that SWAP A$,B$ will cause the string associated with A$ to be assigned to B$ and vice versa. Similarly, SWAP DL,WP will cause the value assigned to DL to be assigned to WP and that of WP to be assigned to DL. If the two variables are not of the same type, (string or numeric) then a 'Type Mismatch' will occur.

The SWAP command saves the need for an intermediate holding variable when exchanging variable values and is considerably faster than this approach. The obvious application is in 'bubble sorts' where elements must be swapped.

A Final Flourish

If you look at Listing 1 around the label RESET you will notice some additional start-up codes. This executes when CoCo does a cold start. The first thing this code does is restore all the drives to track 0. This eliminates that annoying search up and down the disk during the first disk access. The slight increase in start-up time is worth the subsequent savings in access time and reduction in wear and tear on the drive itself, not to mention your nerves!

This code restores all possible drives to track 0. If you do not have four drives you can improve the start-up time a little by only restoring the drives you do have. This is done by changing the '3' in the line immediately after the line defining RESET (which reads 'LDB #3 NUMBER OF DRIVES') to one less than the number of drives you do have.

Adding This Month's Code

Just as in previous months, pull the assembly file built up so far into your editor, then remove the commenting asterisks from the start of line with [REF #] of 2, 9-1, 9-2 and 9-3. Completely delete reference lines 18, 19, 25, 26 and 27. Also delete all lines at the end starting with 'ZZLAST EQU *-1'.

Now type in the new code found in Listing 1 and reassemble the result. As this month's addition is the last, rename the composite assembly language source as *DISKPTCH.ASM* and the binary file as *DISKPTCH.BIN*. Test the binary patch file just as you have for the past few months.

Wrapping It Up Next Month

The next issue of THE RAINBOW will see the last installment of this series. In it we will tie up a few loose ends; put the entire revised version of Disk BASIC in an EPROM and mount it in the disk controller, and make some suggestions for commands you could add yourself. I hope you'll plan on joining me then.

If you would like the entire *DOSPATCH* program source, along with binary files with and without the parallel port driver for *DECB 1.0* and *DECB 1.1*, just send me a disk (no cassettes please) along with $6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly.

Address this request or any questions to: Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.

The listing:

```
                 1087        OPT  LIS
                 1088 *********************************************
                 1089 * PATCH #4 to RSDOS (C)1984 Colin Stearman *
                 1090 *********************************************
                 1091 *
                 1092 * "BAUD" COMMAND CODE
                 1093 * SYNTAX IS BAUD(N) WHERE N =
                 1094 * 300,600,1200,2400,4800,9600
                 1095 *
DC62 BE          1096 BDCNST FCB  $BE,$57,$29,$12,$6,$1 300,600,1200,2400
                 1097                                  4800,9600 BAUD CONSTANTS
                 1098 *
                 1099 *
DC68 BDB262      1100 BAUD   JSR  $B262       EVAL BRKT ARGUMENT
DC6B BDB740      1101        JSR  $B740       GET INTEGER IN X
DC6E 6FE2        1102        CLR  ,-S         FOR COUNTER
DC70 1F10        1103        TFR  X,D         GET BAUD VALUE
DC72 10832580    1104        CMPD #9600       HIGHEST LEGAL VALUE
DC76 1022FBBF    1105        LBHI FCERR       ERROR IF HIGHER
DC7A 6CE4        1106 CNTBD  INC  ,S          COUNT SUBTRACTION
DC7C 83012C      1107        SUBD #300        DIVIDE BAUD BY 300
DC7F 102BFBB6    1108        LBMI FCERR       NOT A VALID VALUE
DC83 26F5        1109        BNE  CNTBD       CONTINUE SUBTRACTION
                 1110 * GOT A VALID MULTIPLE OF 300
DC85 3502        1111        PULS A           GET RESULT
DC87 5F          1112        CLRB             POWER COUNTER
DC88 8EDC62      1113        LDX  #BDCNST     POINT X TO BAUD CONSTANTS
DC8B 44          1114 SFTAGN LSRA             BIT INTO CARRY
DC8C 2503        1115        BCS  GETCON      GOT BIT GET CONSTANT
DC8E 5C          1116        INCB             COUNT SHIFT
DC8F 20FA        1117        BRA  SFTAGN      GO SHIFT AGAIN
DC91 A685        1118 GETCON LDA  B,X         GET BAUD RATE
DC93 9796        1119        STA  BAUDRT      SET BAUD RATE
DC95 0F95        1120        CLR  BDFLAG      CLEAR TO ENABLE SERIAL PORT
                 1121 *                       AND SET LSB OF BAUD RATE
                 1122 *
DC97 39          1123        RTS              ALL DONE
                 1124 *********************************************
```
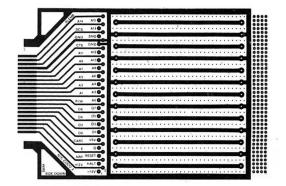
```
                 1125 * "LDIR" COMMAND, PRINT DIRECTORY
                 1126 *
DC98 C6FE        1127 LDIR   LDB  #-2         POINT DEVNUM TO PRINTER
DC9A D76F        1128        STB  DEVNUM
DC9C 7ECBCF      1129        JMP  A0016       DO DIR COMMAND
                 1130 *********************************************
                 1131        IFDF PARPRT      ASSEMBLE FOR PARALLEL PORT
                 1132 *
                 1133 * "PARALLEL" COMMAND CODE AND OUTPUT ROUTINE
DC9F CC01CA      1134 PARA   LDD  #$1CA       120 BAUD DELAY
                 1135 *                       SET MSB TO 1 FOR PARALLEL PORT
DCA2 DD95        1136        STD  BDFLAG      TO MAKE PARALLEL ACTIVE
DCA4 39          1137        RTS
                 1138 **********
                 1139 * Parallel port output routine
                 1140 * This is called by the modified jump at $168
DCA5 0D95        1141 PAROUT TST  BDFLAG      IF NOT ZERO THEN PARALLEL
DCA7 1027EE9F    1142        LBEQ A0015       DO SERIAL OUTPUT
DCAB 3402        1143        PSHS A           SAVE VALUE
DCAD 966F        1144        LDA  DEVNUM      GOING TO DEVICE -2?
DCAF 81FE        1145        CMPA #-2
DCB1 3502        1146        PULS A           RECOVER CHAR, FLAGS DONT CHANGE
DCB3 1026EE93    1147        LBNE A0015       NOT DOING DEVICE #-2
                 1148 *
                 1149 * PARALLEL OUTPUT WANTED
DCB7 810D        1150        CMPA #$0D        WAS IT A CR?
DCB9 2703        1151        BEQ  WASCR
DCBB 0C9C        1152        INC  <$9C        INCREMENT LINE PRINT POSITION
DCBD 8C          1153        FCB  $8C         SKIP NEXT 2 BYTES
DCBE 0F9C        1154 WASCR  CLR  <$9C        LINE COUNTER
DCC0 3411        1155        PSHS CC,X        PRESERVE BASIC VALUES
DCC2 8EFF26      1156        LDX  #DATA       POINT X TO PIA
DCC5 6D1E        1157 CHKRDY TST  -2,X        BUSY IF LINE 7 HI
DCC7 2BFC        1158        BMI  CHKRDY      WAIT UNTIL LOW
DCC9 A784        1159        STA  ,X          DATA REGISTER
DCCB 3511        1160        PULS CC,X        RECOVER VALUES
DCCD 3262        1161        LEAS 2,S         OLD RETURN OFF STACK
DCCF 39          1162        RTS              TO ORIGINAL CALLER
                 1163 *********
                 1164        ENDC
                 1165        OPT  LIS
                 1166 *********************************************
                 1167 * PATCH #5 to RSDOS (C)1984 Colin Stearman *
                 1168 *********************************************
                 1169 *
                 1170 *********************
                 1171 * "SWAP"
                 1172 *
                 1173 * CODE FOR SWAP COMMAND SYNTAX IS SWAP V1,V2
                 1174 * WHERE V1 AND V2 ARE LIKE VARIABLE.TYPES
                 1175 *
DCD0 BDB357      1176 SWAP   JSR  $B357       GET FIRST STRING POINTER
DCD3 9606        1177        LDA  <6          TYPE 0=NUMBER -1=STRING
DCD5 3412        1178        PSHS X,A         SAVE ON STACK
DCD7 BDB26D      1179        JSR  $B26D       PARSE REQUIRED COMMA
DCDA BDB357      1180        JSR  $B357       GET 2ND STRING POINTER IN X
                 1181 * NOW TEST THAT BOTH VARIABLES ARE SAME TYPE
DCDD 3502        1182        PULS A           RECOVER FIRST TYPE
DCDF 9106        1183        CMPA <6          CHECK FOR SAME AS SECOND
                 1184 * NOT SAME TYPE SO ISSUE ?TM ERROR
DCE1 1026D46C    1185        LBNE $B151       TYPE MISMATCH
                 1186 * SAME TYPE SO SWAP POINTER INFO
DCE5 3540        1187        PULS U           ONE IN X, OTHER IN U
DCE7 C605        1188        LDB  #5          COUNTER
DCE9 A684        1189 SWAP5  LDA  ,X          GET VALUE AT X
DCEB 3402        1190        PSHS A           PRESERVE IT
DCED A6C4        1191        LDA  ,U          GET VALUE AT U
DCEF A780        1192        STA  ,X+         PUT AT X
DCF1 3502        1193        PULS A           GET ORIGINAL AT X
DCF3 A7C0        1194        STA  ,U+         PUT AT U
DCF5 5A          1195        DECB             REDUCE COUNTER
DCF6 26F1        1196        BNE  SWAP5       CONTINUE SWAPPING
DCF8 39          1197        RTS
                 1198 ***********************
                 1199 * "ERRORS" Command
                 1200 * Executed when the ERRORS command is encountered
                 1201 *
DCF9 C681        1202 ERRCMD LDB  #$81        CHECK "GO"
DCFB BDB26F      1203        JSR  $B26F       NOT THEN SYNTAX ERROR
DCFE C6A5        1204        LDB  #$A5        CHECK "TO"
DD00 BDB26F      1205        JSR  $B26F       NOT THEN SYNTAX ERROR
DD03 BDAF67      1206        JSR  $AF67       PROCESS LINE # INTO $2B
DD06 DC2B        1207        LDD  <2B         GET THE LINE #
```

```
DD08 DDDC    1208        STD    JLINE    SAVE IT
             1209 ** IF ZERO THEN CLEAR TRAPPING
DD0A 2773    1210        BEQ    ERRSET
             1211 ** CHECK FOR VALID LINE NUMBER
DD0C DCA6    1212        LDD    $A6      GET PARSER POINTER
DD0E 3406    1213        PSHS   D        SAVE ON STACK
DD10 BDAEA9  1214        JSR    $AEA9    CHECK VALID NUMBER
             1215 *IF WE GOT BACK HERE IT'S OK
DD13 3506    1216        PULS   D        RESET PARSER POINTER
DD15 DDA6    1217        STD    $A6
DD17 39      1218        RTS
             1219 ***********************
             1220 *  ERROR TRAPPING AND HANDLING ROUTINE
             1221 *
             1222 * this code is executed when an error is
             1223 * encountered by BASIC from jump at $18F.
             1224 *
DD18 BDD81B  1225 ERRTRP JSR    DIRECT   CURRENT LINE
DD1B 2724    1226        BEQ    NOTRAP   SO DONT TRAP IT
DD1D 9EDC    1227        LDX    JLINE    GET ERRLINE JUMP
DD1F 2720    1228        BEQ    NOTRAP   SO DONT TRAP IT
             1229 ********
             1230 * WE WANT TO TRAP ERROR NOW B HAS ERROR CODE *2
             1231 * IF AN OD ERROR THEN THEN ADDRESS  AT $2B NEEDS
             1232 * PUTTING AT $A6 BECAUSE READ MOVED IT TO SCAN
             1233 * THE DATA STATEMENTS
             1234 *
DD21 C106    1235        CMPB   #$6      OD ERROR NUMBER
DD23 2604    1236        BNE    NOREAD   NOT A OD ERROR
DD25 9E2B    1237        LDX    $2B      GET POINTER
DD27 9FA6    1238        STX    $A6      PUT IT IN PARSER
DD29 54      1239 NOREAD LSRB            DIVIDE BY 2
DD2A D75A    1240        STB    ECODE    CODE ADDRESS
DD2C 9E68    1241        LDX    <$68     CURRENT LINE
DD2E 9F76    1242        STX    ELINE    ERRLINE ADDRESS
DD30 9EDC    1243        LDX    JLINE    GET ERROR GOTO LINE #
DD32 9F2B    1244        STX    <$2B     PREPARE TO GO TO IT
DD34 10DE21  1245        LDS    <$21     CLEAN STACK
DD37 CCADC4  1246        LDD    #$ADC4   RETURN TO INTERPRET LOOP
DD3A 3406    1247        PSHS   D        PUT ONTO STACK
DD3C 0F6F    1248        CLR    DEVNUM   RESET DEVICE CODE
DD3E 7EAEA9  1249        JMP    $AEA9    GO TO NEW LINE
             1250 ****
             1251 *PROCESS NO TRAP
DD41 8D3C    1252 NOTRAP BSR    ERRSET   RESET ERROR CODE
             1253 *
             1254        IFGT   REV      <----
             1255        JMP    $AC49    ; DOS 1.1 only
             1256        ENDC            <----
             1257 *
0000         1258        IFEQ   REV      <----
             1259 * Process new error display  ;
DD43 BDD1E5  1260        JSR    A0026    CLEAR DISK SYSTEM  ;
DD46 3404    1261        PSHS   B        PRESERVE ERROR CODE  ;
DD48 BDCA3B  1262        JSR    A0014    MORE DISK SHUTDOWN  ;
DD4B 3504    1263        PULS   B        GET ERROR CODE BACK  ;
DD4D BDA7E9  1264        JSR    $A7E9    MOTOR OFF  ;
DD50 BDAD33  1265        JSR    $AD33    RESET STACK ETC.  ;
DD53 0F6F    1266        CLR    DEVNUM   REST TO SCREEN  ;
DD55 BDB95C  1267        JSR    $B95C    OUT RETURN IF NEEDED  ;
DD58 54      1268        LSRB            DIVIDE ERROR CODE BY 2  ;
DD59 8D06    1269        BSR    ERFIND   FIND ERROR MESSAGE  ;
             1270 * OUTPUT NEW ERROR MESSAGE  ;
DD5B BDB9A2  1271        JSR    STROUT   OUTPUT IT  ;
DD5E 7EAC65  1272        JMP    $AC65    PRINT " ERROR' ETC.  ; DOS 1.0 only
             1273 *******  ;
             1274 * error message finder  ;
             1275 * B has error count/2 coming in  ;
             1276 * HAS CHARACTER COUNT COMING OUT  ;
             1277 * X HAS POINTER TO FIRST CHAR  ;
DD61 1F98    1278 ERFIND TFR    B,A      MOVE ERROR CODE TO A  ;
DD63 8EDDC5  1279        LDX    #ERR0    POINT X TO MSG #0  ;
DD66 5F      1280        CLRB            DONT AFFECT X FIRST TIME  ;
DD67 3A      1281 KPLOOK ABX             ADD COUNT TO ERROR ADDRESS  ;
DD68 E680    1282        LDB    ,X+      GET CHARS IN MESSAGE  ;
DD6A 4A      1283        DECA            DECREASE ERROR COUNT  ;
DD6B 2AFA    1284        BPL    KPLOOK   KEEP LOOKING  ;
DD6D 39      1285        RTS
             1286        ENDC            <----
             1287 ***************************
             1288 * CLEAR ERROR TRAPPING ON RUN
DD6E BD0F    1289 ERCNCL BSR    ERRSET

DD70 7EC990  1290        JMP    A0013
             1291 ****************************
             1292 * THIS CODE RUNS ON A COLD START AND RESETS ALL
             1293 * DRIVES TO TRACK ZERO AND RESETS ERROR TRAPPING
             1294 *
             1295 * Reset drive 0-1 to track zero
DD73 0FEA    1296 RESET  CLR    $EA      RESTORE OPCODE =0
DD75 C601    1297        LDB    #1       NUMBER OF DRIVES-1
DD77 D7EB    1298        STB    $EB      DRIVE NUMBER
DD79 8D14    1299 NXTDRV BSR    HOME     DO RESTORE TO TRACK 0 WITH 1 RETRY
DD7B 0AEB    1300        DEC    $EB      NEXT DRIVE
DD7D 2AFA    1301        BPL    NXTDRV
             1302 *
             1303 *
             1304 * Clears ERROR trapping
             1305 *
DD7F 3416    1306 ERRSET PSHS   D,X      SAVE REGS
DD81 9E8A    1307        LDX    ZERO
DD83 9FDC    1308        STX    JLINE
DD85 CCFFFF  1309        LDD    #$FFFF
DD88 DD76    1310        STD    ELINE
DD8A 975A    1311        STA    ECODE
DD8C 3516    1312        PULS   D,X      RECOVER REGS
DD8E 39      1313        RTS
             1314 ********
             1315 * restore drive head with no retries
DD8F 3476    1316 HOME   PSHS   A,B,X,Y,U
DD91 8601    1317        LDA    #1       RETRY COUNT 1= NO RETRIES
DD93 7ED670  1318        JMP    A0032    RESTORE CODE ENDS WITH AN RTS
             1319 *********************************
             1320 *
             1321 *    "ELINE"
             1322 *
DD96 DC76    1323 ERRLIN LDD    ELINE
DD98 1083FFFF 1324       CMPD   #$FFFF   IF $FFFF NOT SET?
DD9C 1026FEBD 1325       LBNE   UNSIGN   YES IT IS
DDA0 7EB4F4  1326 SIGNED JMP    $B4F4    RETURN AS SIGNED VALUE (-1)
             1327 ************************
             1328 *
```

```
                    1329 *        "ECODE"
                    1330 *
DDA3 4F             1331 ERRCOD CLRA
DDA4 D65A           1332        LDB    ECODE
                    1333 *IF MINUS THEN IT IS -1 AND THEREFORE UNSET
DDA6 102AFEB3       1334        LBPL   UNSIGN       OUTPUT UNSIGNED # TO VARIABLE
DDAA 1D             1335        SEX                 MAKE D HAVE VALUE IN B
DDAB 20F3           1336        BRA    SIGNED       OUTPUT TO VARIABLE(-1)
                    1337 **************************
0000                1338        IFEQ   REV          <----
                    1339 *      ENAME$  ¦
DDAD D65A           1340 ERNAME LDB    ECODE        GET ERROR CODE        ¦
DDAF 2A02           1341        BPL    GETNM        GET ERROR NAME STRING ¦
DDB1 5F             1342        CLRB                FOR NULL STRING LENGTH ¦
DDB2 A1             1343        FCB    $A1          SKIP NEXT INSTRUCTION  ¦
                    1344 *      ¦
DDB3 BDDD61         1345 GETNM  JSR    ERFIND       RETURNS X AT ERROR NAME ¦
                    1346 *                          B WITH COUNT  ¦
DDB6 1F13           1347        TFR    X,U          SAVE ERROR STRING POINTER ¦
DDB8 BDB50F         1348        JSR    $B50F        CHECK FOR AVAILABLE SPACE ¦
                    1349 *X NOW HAS STRING START ADDRESS ¦
DDBB 2705           1350        BEQ    STREXT       NULL LENGTH STRING ¦
DDBD 1E13           1351        EXG    X,U          SWAP THE POINTERS ¦
DDBF BDA59A         1352        JSR    $A59A        MOVE STRING ¦
DDC2 7EB69B         1353 STREXT JMP    $B69B        RETURN VIA STRING$ CODE ¦
                    1354 ***************************************** ¦
                    1355 *      ¦
                    1356 * ERROR MESSAGES  ¦
                    1357 *      ¦
                    1358 * FORMAT IS CHARACTER COUNT/CHARACTERS ¦
                    1359 *      ¦
DDC5 10             1360 ERR0   FCB    ERR1-(*+1)   ¦
DDC6 4E             1361        FCC    /NEXT WITHOUT FOR/ ¦
DDD6 06             1362 ERR1   FCB    ERR2-(*+1)   ¦
DDD7 53             1363        FCC    /SYNTAX/ ¦
DDDD 14             1364 ERR2   FCB    ERR3-(*+1)   ¦
DDDE 52             1365        FCC    /RETURN WITHOUT GOSUB/ ¦
DDF2 0B             1366 ERR3   FCB    ERR4-(*+1)   ¦
DDF3 4F             1367        FCC    /OUT OF DATA/ ¦
```

```
DDFE 0D             1368 ERR4   FCB    ERR5-(*+1)   ¦
DDFF 46             1369        FCC    /FUNCTION CALL/ ¦
DE0C 08             1370 ERR5   FCB    ERR6-(*+1)   ¦
DE0D 4F             1371        FCC    /OVERFLOW/   ¦
DE15 0D             1372 ERR6   FCB    ERR7-(*+1)   ¦
DE16 4F             1373        FCC    /OUT OF MEMORY/ ¦
DE23 0E             1374 ERR7   FCB    ERR8-(*+1)   ¦
DE24 4E             1375        FCC    /NO SUCH LINE #/ ¦
DE32 09             1376 ERR8   FCC    /SUBSCRIPT/ ¦
DE33 53             1377        FCC    /SUBSCRIPT/ ¦
DE3C 13             1378 ERR9   FCB    ERR10-(*+1)  ¦
DE3D 52             1379        FCC    /REDIMENSIONED ARRAY/ ¦
DE50 0B             1380 ERR10  FCB    ERR11-(*+1)  ¦
DE51 44             1381        FCC    /DIVIDE BY 0/ ¦
DE5C 16             1382 ERR11  FCB    ERR12-(*+1)  ¦
DE5D 49             1383        FCC    /ILLEGAL DIRECT COMMAND/ ¦
DE73 0D             1384 ERR12  FCB    ERR13-(*+1)  ¦
DE74 54             1385        FCC    /TYPE MISMATCH/ ¦
DE81 13             1386 ERR13  FCB    ERR14-(*+1)  ¦
DE82 4F             1387        FCC    /OUT OF STRING SPACE/ ¦
DE95 0F             1388 ERR14  FCB    ERR15-(*+1)  ¦
DE96 53             1389        FCC    /STRING TOO LONG/ ¦
DEA5 12             1390 ERR15  FCB    ERR16-(*+1)  ¦
DEA6 53             1391        FCC    /STRING TOO COMPLEX/ ¦
DEB8 0E             1392 ERR16  FCB    ERR17-(*+1)  ¦
DEB9 43             1393        FCC    /CAN'T CONTINUE/ ¦
DEC7 09             1394 ERR17  FCB    ERR18-(*+1)      DOS 1.0 only
DEC8 46             1395        FCC    /FILE DATA/ ¦
DED1 0C             1396 ERR18  FCB    ERR19-(*+1)  ¦
DED2 41             1397        FCC    /ALREADY OPEN/ ¦
DEDE 0D             1398 ERR19  FCB    ERR20-(*+1)  ¦
DEDF 44             1399        FCC    /DEVICE NUMBER/ ¦
DEEC 0A             1400 ERR20  FCB    ERR21-(*+1)  ¦
DEED 52             1401        FCC    %READ/WRITE% ¦
DEF7 09             1402 ERR21  FCB    ERR22-(*+1)  ¦
DEF8 46             1403        FCC    /FILE MODE/ ¦
DF01 0D             1404 ERR22  FCB    ERR23-(*+1)  ¦
DF02 46             1405        FCC    /FILE NOT OPEN/ ¦
DF0F 15             1406 ERR23  FCB    ERR24-(*+1)  ¦
DF10 52             1407        FCC    /READ PAST END OF FILE/ ¦
DF25 16             1408 ERR24  FCB    ERR25-(*+1)  ¦
DF26 44             1409        FCC    /DIRECT COMMAND IN FILE/ ¦
DF3C 12             1410 ERR25  FCB    ERR26-(*+1)  ¦
DF3D 55             1411        FCC    /UNDEFINED FUNCTION/ ¦
DF4F 0C             1412 ERR26  FCB    ERR27-(*+1)  ¦
DF50 4E             1413        FCC    /NO SUCH FILE/ ¦
DF5C 08             1414 ERR27  FCB    ERR28-(*+1)  ¦
DF5D 52             1415        FCC    /RECORD #/ ¦
DF65 09             1416 ERR28  FCB    ERR29-(*+1)  ¦
DF66 44             1417        FCC    /DISK FULL/ ¦
DF6F 13             1418 ERR29  FCB    ERR30-(*+1)  ¦
DF70 4F             1419        FCC    /OUT OF BUFFER SPACE/ ¦
DF83 0D             1420 ERR30  FCB    ERR31-(*+1)  ¦
DF84 57             1421        FCC    /WRITE PROTECT/ ¦
DF91 09             1422 ERR31  FCB    ERR32-(*+1)  ¦
DF92 46             1423        FCC    /FILE NAME/ ¦
DF9B 09             1424 ERR32  FCB    ERR33-(*+1)  ¦
DF9C 44             1425        FCC    /DIRECTORY/ ¦
DFA5 0B             1426 ERR33  FCB    ERR34-(*+1)  ¦
DFA6 46             1427        FCC    /FILE EXISTS/ ¦
DFB1 0E             1428 ERR34  FCB    ERR35-(*+1)  ¦
DFB2 46             1429        FCC    /FIELD OVERFLOW/ ¦
DFC0 19             1430 ERR35  FCB    ERR36-(*+1)  ¦
DFC1 53             1431        FCC    /SET TO NON-FIELDED STRING/ ¦
DFDA 06             1432 ERR36  FCB    ERR37-(*+1)  ¦
DFDB 56             1433        FCC    /VERIFY/ ¦
DFE1 17             1434 ERR37  FCB    ENDERR-(*+1) ¦
DFE2 41             1435        FCC    /ACCESS PAST END OF FILE/ ¦
DFF9                1436 ENDERR EQU    *            ¦
                    1437        ENDC                <----
                    1438
                    1439
DFF8                1440 ZZLAST EQU    *-1          last used address value
                    1441 *
                    1442 * ZZLAST must not be greater than $DFFF for
                    1443 * DOS 1.0 and $DEFF for DOS 1.1.  The latter
                    1444 * has the OS-9 Boot program and SWI set routines
                    1445 * from $DF00 to $DF4C
                    1446 *
                    1447 *
                    1456        OPT    LIS
D994                1457        END    ADDCOM
```

High Fidelity Hardware Hacking

By Ed Ellers
Rainbow Technical Writer

● *Is there any way to get parallel data in and out of a CoCo? I would like to store data using an eight-track tape recorder at a higher speed than my single track serial data recorder. Would I need more than eight tracks?*

Is it possible to change the Baud rate of data to and from the cassette recorder like you can to the printer? I use a Pioneer reel-to-reel tape deck, which I'm sure can handle a faster Baud rate.

David DeGomez
Denver, CO

The idea of using an eight-track recorder to store data has been tried before, but not in the way you describe. It sounds like you're talking about recording eight tracks simultaneously; that is not what the conventional eight-track tape decks do, and would require a special recording head. As for faster cassette I/O, you might try this if your computer will accept the speed-up poke:

```
POKE 65497,0:CSAVE"filename"
:POKE 65496,0
```

(Ed Ellers, a RAINBOW *and* PCM *staff member, is a self-confessed electronics fanatic whose other interests include science fiction.)*

This should *CSAVE* a program at 3000 Baud instead of 1500, though I make no promises. The reason that I have the two *POKEs* in a single command line is that *POKEing 65497* disrupts a number of normal CoCo functions (including video); this method does the *POKE,* saves the file and then puts the machine back at normal speed. The same procedure (with *CLOAD* in the middle) would be used to load such a file. You shouldn't need to move up to a different recorder to do this.

Frozen Contacts

● *I have a General Electric cassette recorder for my CoCo, and all of a sudden the recorder won't stop when it's done loading or recording a program. A friend said that my computer is not turning the recorder off and it is stuck in the "on" position. Is there any way to fix this without opening up the computer?*

D. Milsom
Newark, DE

I don't see how. The problem is probably in the relay inside the CoCo that controls the cassette motor; it may have

become spot-welded in the closed position from excessive current and the resulting sparks. Early models of the CoCo (C, D and E boards) had a rather small relay; newer machines had a better one that clicks rather loudly. I'd have to say your recorder probably contributed to the problem. Radio Shack recorders are wired so that the remote jack switches only the motor; most other makes (including GE, as far as I know) switch all power within the recorder, which means that the relay has to carry more current than Radio Shack allowed for when they picked that relay for the CoCo. To make absolutely sure that the CoCo is at fault before trying a repair, unplug the remote plug from the recorder and connect an ohmmeter across the plug contacts. The remote circuit should read "open" unless you have the computer in a cassette I/O function (or you have selected *AUDIO ON*).

Moving The Multi-Pak

● *I received a Multi-Pak Interface for my CoCo as a Christmas gift. It seems to work properly when hooked directly to the expansion port on my computer. The problem, however, is that my computer desk is similar to a typing desk and*

there isn't room for both the computer and the interface.

To solve the problem I constructed an interface cable about 38 inches long. I brought all 40 lines and the two grounding tab lines out of the computer to the Multi-Pak, but it doesn't work. The wiring is known to be correct, because the cable works fine for the disk controller and various cartridges. I tried a shorter (12-inch) cable with the same results. I then tried to shield and ground the whole cable with aluminum foil, and this helped, but the cable still wasn't reliable. Would you have any knowledge of a solution to this problem?

John J. Gallagher
Williamson, NY

The Multi-Pak Interface doesn't seem to be able to work with extender cables, because of the many signals that go back and forth between the CoCo and the Multi-Pak unit. You might try using a commercially available shielded or twisted-pair cable (available from industrial electronics distributors), but I'm hesitant to tell you to spend the money because it may well do no good whatsoever.

Printer Echo No Go

• *I read with much anticipation the hardware article dealing with connecting the 4-pin DIN plugs of the printer and modem together, so that everything appearing on the screen would be printed. I did the whole project, and then discovered that the Radio Shack DMP-100 printer doesn't have a 300 Baud setting. Is there any way to change the Baud rate setting on the DMP-100 to 300 either by software or by a hardware project?*

Craig Luecke
Brinklow, MD

The DMP-100's two Baud rates are fixed in the ROM inside the printer, and changing to 300 would be rather difficult. You could use one of the several serial-to-parallel interfaces on the market to pick up 300 Baud data and feed it into the DMP-100's parallel port. Note that this will only work if the system on the other end is operating at 8 bits with no parity; if 7 bits with even or odd parity is used there will most likely be garbage on the printer, because the characters with the parity bit added will correspond to special characters instead of regular alphanumerics.

---

32 or 64 — What's The Difference?

• *I have a 32K CoCo. I hear so much about having 64K, but I wonder what the difference is between my computer and the 64K model.*

Also, I have a short program that moves BASIC into RAM. Does that make my computer 64K?

Michael Rosenberg
Prestonsburg, KY

If your BASIC-in-RAM program works, then that means the machine in question is actually 64K. Radio Shack made a large number of machines between late 1982 and the middle of 1983 that actually had 64K of RAM; they only advertised them as 32K because none of the software they sold (until OS-9 became available) could make use of the all-RAM 64K mode. Some other machines were sold with 64K RAM chips installed, but were only wired for 32K because Tandy saw no need to provide 64K capability at the time.

Ed Ellers will try to answer any questions you may have about the CoCo and its applications. Send your questions to "Earth To Ed," THE RAINBOW, P.O. Box 385, Prospect, KY 40059.

(Continued from Page 16)

Too long	2.6
Too many	1.8
Too few	7.2
Too generous	10.7
Too critical	0.3

Do our reviewers seem:

Well qualified	37.0
Qualified	60.0
Under qualified	3.0

Do you wait to see a RAINBOW review before buying a product?

Always	24.3
Usually	51.8
Sometimes	22.3
Never	1.6

Are you hesitant to buy from a new advertiser?

Yes	57.4
No	42.7

Are you more inclined to buy after seeing a product advertised?

Once	12.6
Two or three times	65.0
Six times	13.4
More than six times	9.0

Do you like the discount coupon pages?

Yes	85.7
No	14.3

Do you like our issue themes?

Yes	93.2
No	6.8

Do you use RAINBOW Check?

Frequently	18.4
Seldom	35.2
Never	46.4

Would three-column listings of BASIC programs (in smaller type) be useful?

Yes	36.3
No	63.7

Do you get RAINBOW ON TAPE?

Yes	35.3
No	64.7

Age:

18 or Under	17.2
19-25	8.6
26-35	22.3
36-45	29.0
46-65	20.2
Over 65	1.7

Occupation:

Engineer/Technical	27.6
Professional	19.8
Business person	18.7
Student	18.3
Educator	7.5
Skilled Trade	5.0
Retired	3.0

Household Income:

Under $10,000	9.2
$10,000-$20,000	10.1
$21,000-$30,000	21.1
$31,000-$50,000	38.2
$51,000-$75,000	15.2
Over $75,000	6.2

We had some other questions dealing strictly with the show itself and a few questions were so ambiguous that the response was statistically meaningless. For instance, there was hardly any numerical spread to the "rate our departments" and "rate our themes" responses; this is probably due to the poor wording of the questions, but could possibly be because the respondents had a wide disparity of opinions, each with their own favorites. In general, though, the results suggest that our columns on BASIC, letters to the editor and technical Q&A columns are most highly rated. RAINBOW Scoreboard and its accompanying Scoreboard Pointers were at the bottom. In regard to themes, apparently the most popular are graphics, hardware and data communications, while the lowest ratings went to our holiday (bah, humbug, yourself), music and Simulations. With more specific questions, we should be able to learn more when we poll those who attend the Irvine, Calif., RAINBOWfest. Then, maybe we'll be ready to poll the entire readership. In the meantime, a reminder that subscription rates are being slightly increased and that you can save money by subscribing, or extending your subscription, before the first of the year.

— Jim Reed

RAINBOW

Give us your best: Join the ranks of these courageous CoCoists in showing the Color Computer world your high score at your favorite micro-diversion. We want to put your best effort on record in THE RAINBOW's Scoreboard column. All entries must be received by the first of the month to be eligible for the following month's Scoreboard. They must include your *full* name, address, game title, company name and, of course, your high score. Each individual is limited to three score entries per month. Send your entries to Scoreboard, c/o THE RAINBOW.

★ **New Number One**

ANDRONE *(Radio Shack)*
40,030	★Matthew W. Swadling, North Vancouver, British Columbia	
27,805	John Marcogliese, Eastchester, NY	
27,450	Alan Bucior, Victoria, British Columbia	
18,290	Bill Sain, Charlotte, NC	
17,170	Jamie Schultz, Ontario, Canada	

AREX *(Adventure International)*
29,794,900 ★Barbara Blumenberg, St. Francis, WI

BAG-IT-MAN *(Aardvark)*
418,790	★Cornelius Caesar, Hofheim, West Germany
310,400	Eric Lecrouart, Ottawa, Ontario
109,620	Ronald Gates, Grand Rapids, MI
101,400	Daniel Belisle, Montreal, Quebec
70,510	Stephane Asselin, Hauterive, Quebec

BASEBALL *(Radio Shack)*
464-0	★Dave Fisher, St. Downey, CA
460-0	Walter Trainlips, Janesville, WI
324-0	Michael Rosenberg, Prestonsburg, KY
297-0	Joe Milligan, Julian, PA
296-0	Seth Newman, Yardley, PA
136-0	Stefan Mecay, Austin, TX
103-0	Rusty Johnson, Des Moines, IA
100-0	Chris Dunlap, Springfield, MA
40-0	Mario Asselin, Hauterive, Quebec

BATS AND BUGS *(THE RAINBOW)*
24,600	★Michael Rosenberg, Prestonburg, KY
7,200	Lezlee Bishop, Salt Lake City, UT
3,600	Apollo Latham, Rich Square, NC
3,350	Eugene Watanabe, Salt Lake City, UT
3,300	Joey Lewis, Cabin Creek, WV

BUZZARD BAIT *(Tom Mix)*
6,447,950	★Jon Griffith
5,488,250	Jim Kennett
2,983,350	Blossom Mayor, East Greenwich, NY
2,902,700	Michael Popovich, Nashua, NH
2,087,650	Edmund Greene, Nashua, NH

CANYON CLIMBER *(Radio Shack)*
8,990,000	★Glen Giacomelli, Woodbridge, Ontario
2,326,200	Scott Oberholtzer, Lexington, MA
1,603,400	Shen Mansell, Calgary, Alberta
1,571,300	Jeff Weaver, Gordonville, PA
1,426,600	Sean Whitley, Arvada, CO
512,000	Lisa Veaudry, Midland, Ontario

CASHMAN *(MichTron)*
$25,320	★Chris Dunlap, Springfield, MA
$20,750	Melvin Pettuk, Nampa, Alberta
$19,330	Dan Ralston, Janesville, WI

CAVERN COPTER (THE RAINBOW)
1213	★Doug Schwartz, Glendale, AZ
968	Michael Mefferd, Wren, OH
790	John Rivers, North Adams, MA
760	David Figel, Sardis, OH
747	Susan Ballinger, Uxbridge, Ontario

CHOPPER STRIKE *(MichTron)*
162,400	★Andrew Figel, Sardis, OH
130,200	Benny Pischke, Lloydminster, Saskatchewan
87,600	David Figel, Sardis, OH
83,300	Doug Masten, Macon, OH
72,100	Lisa Siclari, Staten Island, NY

CLOWNS & BALLOONS *(Radio Shack)*
143,300	★Moe Tindell, Sebring, FL
116,495	Jeffrey Kochs, Grove City, OH
116,470	Colin Kerridge, Ladysmith, British Columbia
114,140	Cheryl Pratt, Moab, UT
110,475	Andrew Truesdale, Ferguson, MO

COLOR BASEBALL *(Tandy Corporation)*
61-0 ★Paul Brouwers, Ontario, Canada

COLORPEDE *(Intracolor)*
10,001,051	★Mark Smith, Santa Ana, CA
5,756,808	Scott Oberholtzer, Lexington, MA
3,355,248	Scott Drake, Pine City, NY
2,614,230	Jerry Petkash, Warren, MI

2,547,299   Rich McGervey, Morgantown, WV

CRUISING *(Sunshine)*
3,936   ★Paul Hotz, Herzlia, Israel

DEFENSE *(Spectral Associates)*
48,350	★Jeff Kennedy, Scottsburg, IN
40,000	James Kennedy, Scottsburg, IN
37,000	Kevin Hawkins, Scottsburg, IN

DEVIL ASSAULT *(Tom Mix)*
4,569,305	★David Fitzpatrick, Ogallala, NE
3,096,900	Chad Barry, Rochester, NH
3,048,400	Brent Murphy, Mesa, AZ
2,890,000	Rich Van Manen, Grand Rapids, MI
1,762,980	Michael Rosenberg, Prestonsburg, KY

DOWNLAND *(Radio Shack)*
38,954	★Mary Ward, Groton, CT
35,539	Rusty Johnson, Des Moines, IA
23,000	Dan Feinstein, Mobile, AL
23,000	Doug Feinstein, Mobile, AL
22,304	Don Ward, Groton, CT
19,427	Albert Diaz, Canoga Park, CA

DRACONIAN *(Tom Mix)*
395,400	★Peter Kral, Arcata, CA
326,180	George Hoffman, Shavertown, PA
190,840	Kyle Keller, Overland Park, KS
139,010	Paul MacArthur, Gillette, WI
181,710	Bryan Jenner, Calgary, Alberta

DUNKEY MUNKEY *(Intellectronics)*
2,116,200	★Dan Price, Tacoma, WA
1,936,000	Tim Greenen, Sterling Hts., MI
1,244,400	Jack Baran, Bensalem, PA
1,015,000	Kyle Keller, Overland Park, KS

FIRE COPTER *(Adventure International)*
112,550	★George Dougherty, Hampton, VA
107,370	Chris Hafey, Auburn, CA
97,390	Sam Hughes, Colton, CA
78,860	Woody Farmer, Acme, Alberta
53,280	Kevin Marsh, Bokeelia, FL

FOODWAR *(Arcade Animation)*
104,075	★Mike Scharf, Fremont, OH
94,905	Stephane Asselin, Hauterive, Quebec

FROGGIE *(Spectral Associates)*
86,660	★David Garozzo, Morrisville, PA
84,440	Bill Ide, Newark, DE
74,050	Mike Garozzo, Morrisville, PA
73,190	Brendan Smith, Coral Springs, FL

GONE FISHING (THE RAINBOW)
29	★Eric Burk, Williamsvill, NY
12	Lisa Ballinger, Uxbridge, Ontario
12	Kevin Oberberger, Sparks, NV
11	Brian Austin, New Salisbury, IN
11	Emily Doubt, Deep River, Ontario

GRABBER *(Tom Mix)*
147,600	★Brian Foley, Blackstone, MA
129,100	Blossom Mayor, East Greenbush, NY
70,600	Michael Corman, Lafayette, IN
31,900	Stephane Asselin, Hauterive, Quebec
27,750	Ellen Ballinger, Uxbridge, Ontario

HEIST (THE RAINBOW)
2,100	★Sergio Waisser, Mexico City, Mexico
1,500	Susan Ballinger, Uxbridge, Ontario
1,500	Julio Comello, Scarborough, Ontario
1,500	Kirstie Compton, Suffield, CT
1,500	Andy Dater, Medford, OR
1,500	David Figel, Sardis, OH
1,500	Tim Hoven, Eckville, Alberta
1,500	Joel Lombardi, Newark, DE
1,500	Jason Morrison
1,500	Jeff Roberg, Winfield, KS
1,500	Brendan Smith, Coral Springs, FL
1,500	Kevin Speight, Bridgewater, Nova Scotia
1,500	Rupert Young, Sheffield, MA

HYPER ZONE *(Computerware)*
59,830   ★Mario Asselin, Hauterive, Quebec

INVASION *(Spectral Associates)*
88,000	★Kevin Hawkins, Scottsburg, IN
35,420	Jeff Kennedy, Scottsburg, IN

THE JUNGLE (THE RAINBOW)
1,110,729	★Tony Boring, Armagh, PA
644,694	Matt Hazard, Columbia Station, OH
29,735	Craig Boring, New Florence, PA
4,230	Doug Schwartz, Glendale, AZ
3,048	Jon Hobson, Plainfield, WI

KATERPILLAR ATTACK *(Tom Mix)*
65,682	★Jason Morrison
38,676	James Lafave, Williamstown, MA
31,672	Scott Fairfield, Williamstown, MA
18,949	Vadim Gotovsky, Toronto, Ontario

LANCER *(Spectral Associates)*
74,250   ★Matthew Crabtree, Pittsburgh, PA

LUNAR—ROVER PATROL *(Spectral Associates)*
79,950	★Paul Hotz, Herzlia, Israel
45,600	Bryan Jenner, Calgary, Alberta

MAZELAND *(Chromasette)*
5,500   ★Todd Knapp, Prairie du Chein, WI

MICROBES *(Radio Shack)*
178,550	★Apollo Latham, Rich Square, NC
144,350	Theodore Latham Jr., Rich Square, NC
98,450	Kent Jakway, Garrett, IN
59,960	Steven Allen, Sharpsburg, MD
40,850	Joey Lewis, Cabin Creek, WV

MOON HOPPER *(Computerware)*
114,540	★Susan Ballinger, Uxbridge, Ontario
78,820	Brian Austin, New Salisbury, IN
59,970	Cole McDonald, St. Cloud, MN
58,520	Brendan Smith, Coral Springs, FL
53,570	Robert Harmon, Virginia Beach, VA

MR. DIG *(Computerware)*
2,520,650	★Thomas Henry, Boca Raton, FL
2,371,550	Ann MacKery, Port Hardy, British Columbia
2,301,000	Jeff Roberg, Winfield, KS
1,976,500	Tim Magnusen, Lafayette, TN
1,881,350	Ellen Ballinger, Uxbridge, Ontario

MUDPIES *(MichTron)*
365,900	★Glenn Wasson, Castleton, NY
185,200	Bertha Jeffries, San Bernardino, CA
181,200	Anthony van Dijk, The Pas, Manitoba
173,900	Andrew Chin, Austin, TX
164,000	Paul Baker, Pittsburgh, PA
77,100	T.J. Buckley, Warrenton, VA

NINJA WARRIOR *(Programmer's Guild)*
157,400	★Sandra Maguire, Port Elgin, Ontario
151,100	Douglas Rodger, Harvard, MA
106,300	Bud Seibel, Tumbler Ridge, British Columbia
105,200	Martin W. Grimm, Elkview, WV
102,400	Christopher Gelowitz, Claresholm, Alberta

PAC 'EM (THE RAINBOW)
2,372	★Stephanie Gregory, Coco Solo, Panama
2,301	Lawrence Gore, Glencoe, AL
2,106	Dr. James Peterson, Radcliff, KY
2,059	Art Hartsough, Ft. Wayne, IN
1,999	Kevin R. Hubbard, Huntington, WV

PARACHUTE JUMP *(Jarb Software)*
62,650   ★Dan Sobczak, Mesa, AZ

PARA-JUMPER (THE RAINBOW)
3,118	★Jon Clevenger, Lima, OH
822	Peter MacLeod, Montague, Prince Edward Island
783	Ronald Gates, Grand Rapids, MI

PIPELINE (THE RAINBOW)
478	★Susan Ballinger, Uxbridge, Ontario
302	Lisa Ballinger, Uxbridge, Ontario

POPCORN *(Radio Shack)*
64,380	★Susan Rushing, Tucson, AZ
57,860	Jeffrey Kochs, Grove City, OH
54,280	Wyatt Godines, Lawrence, MI
48,930	Paul Baker, Pittsburgh, PA
48,520	Nicole Freedman, Wellesley, MA
39,030	Susan Ballinger, Uxbridge, Ontario
23,540	Teri Block, Sewell, NJ

SCOREBOARD

PYRAMID *(Radio Shack)*
220/113	★John Dupre, Mobile, AL
220/130	Cornelius Caesar, Gundelhardtstr, West Germany
220/130	George R. Fairfield, Victoria, British Columbia
220/133	Robert Dickau, Sacramento, CA
220/136	Andy Nelson, Winona, MN
220/138	Ellen Ballinger, Uxbridge, Ontario

Q-NERD *(THE RAINBOW)*
6,512,020	★Ray Ravalitera, Bethune, France
326,810	David Chabot, Granby, Quebec
184,780	Ray Suplee
181,920	Susan Bennington, Pensacola, FL
130,000	Robert Dickau, Sacramento, CA

RAAKA-TU *(Radio Shack)*
50	★Mike Bubb, Grafton, OH
50	★Chris Cope, Central, SC
50	★Richard Gain, Tyler, TX
50	★John Kidd, Clarksboro, NJ
50	★Barney Laverty, Gauley Bridge, WV
50	★Michael Thomas, Flint, MI
40	Ellen Ballinger, Uxbridge, Ontario
20	Russ Auld, St. Petersburg, FL

RADIO BALL *(Radio Shack)*
1,103,090	★Mike Sengstock, Meriden, CT

REACTOIDS *(Radio Shack)*
3,781,815	★Rusty Johnson, Des Moines, IA
931,395	Linda Mobbs, Pt. Huron, MI
702,125	Jeff Loeb, Mobile, AL
203,800	Andrew Lehtola, Mound, MN
88,615	Robbie Anderson, Monrovia, CA

RETURN OF THE JET-I *(ThunderVision)*
406,788	★Chris Russo, Miami, FL
395,029	Joseph Russo, Miami, FL
389,453	Gary Bachtel, Huntsville, AL
208,602	Robert Buerkle, Conway, PA
188,000	Todd Kaplan, Lawrenceville, NJ
65,050	Todd Knapp, Prairie du Chein, WI

ROBOTTACK *(Intracolor)*
7,528,100	★Mike Scharf, Fremont, OH
3,242,812	Edwin Prather & Cory Soper, Oxnard, CA
2,516,050	Horace Hamilton, Calgary, Alberta
2,216,950	Randy Hankins, Tabor, IA
1,922,200	Erik Merz, Noblesville, IN

SCOOPER PACK *(Spectral Associates)*
5,000,000	★Jeff Kennedy, Scottsburg, IN
2,400,020	Kevin Hawkins, Scottsburg, IN

SEA QUEST *(Mark Data Products)*
100/147	★Harry Price, Tacoma, WA

SHOOTING GALLERY *(Radio Shack)*
149,940	★Robert Wallace, Waldorf, MD
85,120	Lezlee Bishop, Salt Lake City, UT
80,220	Vernell Peterson, Radcliff, KY
44,870	Mark Nichols, Birsay, Saskatchewan
44,480	R. Duguay, St. Bruno, Quebec

SKIING *(Radio Shack)*
01.10	★Mike Scharf, Fremont, OH
05.85	John Hokpins, Greenville, SC
12.02	Brian Austin, New Salisbury, IN
12.08	Kelly Kerr, Wentzville, MO
12.58	Kevin Hawkins, Scottsburg, IN
13.35	Jeff Kennedy, Scottsburg, IN

SKRAMBLE *(Tom Mix)*
61,020	★Paul Hotz, Herzlia, Israel
46,440	Steve Schutjer, Hazel Green, WI

SLAY THE NEREIS *(Radio Shack)*
344,115	★Ed Meyer, Vancouver, British Columbia
47,442	Joseph DiBattista, Brooklyn, NY
34,089	Craig Alexander, Heflin, AL

SNAKER *(THE RAINBOW)*
1:26	★Dan Sobczak, Mesa, AZ
1:57	Susan Ballinger, Uxbridge, Ontario
1:59	Baiju Shah, Deep River, Ontario
2:21	Eric Burk, Williamsvill, NY

STARBLAZE *(Radio Shack)*
11,000	★Steve Schutjer, Hazel Green, WI
9,950	Albert Diaz, Canoga Park, CA
9,700	Robbie Sablotny, Mt. Zion, IL
9,050	Mark Welte, Baxter, TN
6,850	Craig Alexander, Heflin, AL

STELLAR LIFE-LINE *(Radio Shack)*
100,070	★Alex Gatlin, McKinleyville, CA
33,100	Kenn Booth, Grand Rapids, MI
29,670	Craig Alexander, Heflin, AL
22,610	David Sullens, Cabool, MO
16,920	Jeff Loeb, Mobile, AL

STORM ARROWS *(Spectral Associates)*
35,000	★Stephane Asselin, Hauterive, Quebec

TIME BANDIT *(MichTron)*
413,620	★Doug Seibel, Tumbler Ridge, British Columbia
248,530	Sally Naumann, Hailey, ID
243,620	Mark Wooge, Omaha, NE
225,950	Chris Oberholtzer, Lexington, MA
142,200	Fred Naumann, Hailey, ID

TOUCHSTONE *(Tom Mix)*
98,500	★Glenn Wasson, Castleton, NY
88,300	Michael Mefferd, Wren, OH
65,520	Kevin Marsh, Bokeelia, FL

TRAILIN' TAIL *(THE RAINBOW)*
105,300	★Jerry Dill, Frankfort, MI
87,345	Philip Parent, Smiths Falls, Ontario
76,275	Michael Rosenberg, Prestonsburg, KY
74,395	Jean-Marc Parent, Smiths Falls, Ontario
33,454	Kenneth Bergenham, Lawton, MI

TUT'S TOMB *(Mark Data)*
163,060	★Michael McCafferty, Oceanside, CA
158,000	Chris Russo, Miami, FL
106,460	Eileen Kaakee, Royal Oak, MI
104,360	Gary Marshall, Layton, UT
60,780	Stephane Asselin, Hauterive, Quebec
27,320	Chris Dunlap, Springfield, MA

WACKY FOOD *(Arcade Animation)*
241,200	★Todd Kaplan, Lawrenceville, NJ
227,900	Jon Jenkins, Milner, GA
135,800	Mario Asselin, Hauterive, Quebec
105,100	Stephane Asselin, Hauterive, Quebec

WILDCATTING *(Radio Shack)*
250,450	★John Kidd, Clarksboro, NJ
33,090	Jack Ballinger, Uxbridge, Ontario

ZAXXON *(Datasoft)*
2,057,800	★Chris Oberholtzer, Lexington, MA
1,510,000	James Quadrella, Brooklyn, NY
666,000	Andy Green, Whitehall, PA
401,900	Mike Hughey, King George, VA
370,400	Chris Coyle, Selden, NY
150,050	Jeff Kennedy, Scottsburg, IN
135,000	Kevin Hawkins, Scottsburg, IN
102,000	Bryan Jenner, Calgary, Alberta
74,700	Mike Sengstock, Meriden, CT

Debbie Hartley

SCOREBOARD POINTERS

In conjunction with THE RAINBOW's Scoreboard, we offer this column of pointers for our game-playing readers' benefit. If you have some interesting hints and tips, we encourage you to share them by sending them to the Scoreboard, c/o THE RAINBOW.

PROGRAM EXCHANGE

Scoreboard:

Anybody looking for another insane CoCo user to trade programs with? I would appreciate it very much. It would make my gloomy day!

Todd Knapp
Prairie du Chien, WI

DRAIN PROBLEMS?

Scoreboard:

This is the best magazine that has ever been put out on the CoCo. It takes all the bits of information from a variety of sources and compiles it into one. Keep up the good work.

I have several Adventure games and the one that gives me the most problem is *Sands of Egypt*. I would appreciate it greatly if anyone can help out. I've found the pool, gotten in it, can't get out, and I know I have to drain it somehow. I've also found the big palm tree. Please send the solution if possible. My address is 51-2 Maine St., 94535.

Jeff McKay
Travis A.F.B., CA

THE 220 POINT QUESTION

Scoreboard:

In the game *Pyramid,* what happens at 220 points? Please write to let me know. My address is 501 Buckley Highway, 06076.

Chris Caovette
Union, CT

IN LIMBO

Scoreboard:

Help! I am having trouble with *Calixto Island.* What do you do after you go down the trap door? I can't get my flashlight on. Also, in *Black Sanctum,* I have found the beach house, but I can't do anything with it. My son and I will be waiting for your

replies so we can get going. Write me at 1446 Marline Ave, 92021.

Liz Newman
El Cajon, CA

MISSION IMPOSSIBLE?

Scoreboard:

As it is neither practical nor possible for your editors to verify the legitimacy of your readers' score submissions, you cannot be criticized for publishing them, verbatim, on the assumption that they are genuine. No doubt, the possibility of deceit exists in any honor system.

Most (if not all) arcade-type games allow for phenomenally high scores. In contrast, logical (Adventure) games generally do not. Adventure games usually have a fixed, maximum number of point-scoring objects, and require a certain minimum number of moves to realize a particular score. Moreover, whereas someone who has achieved an unprecedented high score in a "shoot-em-up" game probably cannot duplicate that score on demand (perhaps never again), one who has reached an optimal solution in a determinate game should be able to reproduce that score consistently. With the exception of a trivial quasi-random element, the Adventure game *Pyramid* is a determinate game — solutions are replicable.

I do not believe that it is possible to score 220 points in *Pyramid* in less than 130 moves. The proof relies upon details of the program's (sloppy, but copyrighted) source code and data, and so will not be given here, but it is believed to be accurate and complete.

George R. Fairfield
Victoria, British Columbia

BEST BETS FOR BEDLAM

Scoreboard:

For those of you having trouble with *Bedlam*, pay attention! Get the red key in the Dispensary, but first go to the Maintenance room and get the hook. Go back to the Dispensary and type GET RED KEY WITH HOOK. If the man with the hypo gets you just type PLUGH. Go right outside the shock room and type GET GREEN KEY WITH HOOK. Now that you have everything you need, find Napoleon. Go to every room and type OPEN DOOR. Pretty soon you will find a secret door. You will not be able to open it, so ask Napoleon to open it and you're home free. I forgot to mention that in order to get Napoleon to follow you, type ASK NAPOLEON TO FOLLOW.

In *Raaka-Tu*, search each room by typing SEARCH ROOM. You should find a gem in a crack. That's 10 points right there. Also, you should never pull the lever, just get it.

In *Dungeons of Daggorath*, I think I have the list of rings:

NAME	INCANT TO
Vulcan	Fire
Lime	Ice

John Kidd
Clarksboro, NJ

CHOCK FULL O' HINTS

Scoreboard:

Here are some tips on *Black Sanctum, Calixto Island, Seaquest,* and *The Crown of Merro.*

To get past the natives on *Merro*, type LIGHT MATCH. Then type GIVE MATCHES. This will get you past them. To get past the poisoned arrows you must have the water. (You get this by pushing the nose on the fountain.) Type DRINK WATER. I'll give one more hint; you had better have your hacksaw before going in any further!

To get the caretaker's tools on *Sanctum* just give him the jug of wine.

On *Calixto*, always get the boots. To get the paddles you must have the mice. To get them you need the trap. Then go to the pyramid and DROP MICE when you get to the snake. Always remember to have the bucket with you when you cross the river!

On *Seaquest*, to get into the beach house you must break the window. (The beach house is behind the falls. Just type GO FALLS and you will be in the cavern. When you enter the cavern you will trip over something. That is the shovel. Type GET SHOVEL.) To open the clam you must have the shovel. To get to the clam you must buy the air tanks with the credit card. Then type FILL TANKS. Now you can go into the water.

If anybody has found a way to open the trapdoor in the beach house or has found a way to buy the speargun on *Seaquest*, then write to me at 390 St. Charles Ave., 32953.

Keith Schuler
Merritt Island, FL

PRESS AHEAD

Scoreboard:

I would like to pass along to the readers a little hint to use while playing *The Frog* from Tom Mix. I found out that on my CoCo 2 (64K ECB) the frog would only go once in any direction. Meaning that in order to go across the street you had to first press the up-arrow once, then go left or right before you could go ahead once more. I found that by holding down the space bar you could travel in any direction more than once. So by doing this, you can

travel across the street or logs more than once just by pressing the up-arrow. If anyone knows of a software fix for this I am sure the readers that have this program would like it.

John D. Cleveland
Lunenburg, Nova Scotia

Scoreboard:

The game *Arex* has 32 different levels. After 32 it starts repeating itself from Level 12. At Level 60 you do not start out with any stars, but they do come into play. At Level 256 the game starts over with Level 1 and one star, and then increases the stars with the point value. The score only goes to 700,000 and I went through that 42 times.

This game also throws distractions in. When you get to the higher levels it will go slow and then all of a sudden it will speed up again. At times during the game it takes all your men away and you have to start over, but by then each square is worth so much that·in a couple of moves you have one back.

The only help I can give in playing the game is to play aggressively. Go right after the stars. When you have 12 at a time you can usually get about 10 before they get you, if you go after them.

Barbara Blumenberg
St. Francis, WI

Scoreboard:

Enough is enough! I need help with *Sands of Egypt.* A recent pointer in the Scoreboard hinted that you need to place the scepter on the sarcophagus to get into the treasure room.

I know this is a dumb question, but how do you place the scepter on the sarcophagus? I can't even drop the darn thing!!

Send any help to P.O. Box 115, 20862. Thank you.

Craig Luecke
Brinklow, MD

Scoreboard:

I have solved the following Adventure games: *Dungeons of Daggorath, Pyramid, Arconiax Assignment, Bedlam, Calixto Island, Karrak, Crown of Merro,* and *The Trip.* If you need help with one of these Adventure games send a SASE and I will reply.

If anyone has solved *Pyramid 80*, I need help.

Chuck Poynter
Hector, AR

RAINBOW REVIEWS

HARDWARE

DOUBLE SWITCH — Two LEDs show you which port is being used, 1 or 2. High Quality parts with a great looking face plate! $29.95

DOUBLE CABLE — Hook a MODEM and a PRINTER up at the same time with this Y-connector $14.95

EXT-CABLE — long printer or MODEM cable (15 feet) $14.95

Y-CABLE — You can connect two devices at the same time to your ROM port (80 column card and disk Drive) $29.95

DOUBLE DRIVER — Best video driver available for your CoCo. Made by our friends at Moreton Bay Software. Specify CoCo or CoCo II $24.95

MINI MOUTH — Now get sound from your mute monitor. Plugs right in, nothing to solder $24.95

COLOR POWER II — This plug in CP/M board will allow you to use thousands of CP/M programs $329.00

MINI-MODEM — 300 BAUD, Originate/Answer Full Duplex, Direct Connect . $79.95 (Save $10.00 when you buy one of our terminal programs and a modem)

DISKS — DISKS — DISKS — DISKS — DISKS
Save on blank 5¼'' diskettes. Buy in bulk and save! No sleeves. (10 minimum in each order) $16.00

DOUBLE SWITCH

DOUBLE CABLE

DOUBLE DRIVER

MINI-MOUTH

ULTRA TERM + Plus +

This program is the **ultimate** in CoCo communicating!! Ultra Term+ is used with a plug-in 80 column board. Supports either Double 80 Plus, Color Power II or Word Pak.

Here are just some of the features Ultra Term+ has to offer:

Select:
  Half, Full Duplex or Echo
  Odd, Even, Mark, Space or No Parity
  7 or 8 Bit Words
  1 or 2 Stop Bits
  All Caps if needed
  Several Printer Formats
  Trapping of incoming characters

BAUD Rates:
  110-4800 (communicate)
  600-9600 (printer)

Screen Format:
  80 x 24 upper/lowercase
  Send all 128 characters from keyboard

Buffer:
  Merge text or programs
  49K to 53K memory
  Four Buffer Send Modes
  Display Bytes Used/Remaining
  Editor—Move forward and reverse thru buffer. Insert, type over, delete lines, characters or words. Block delete.

10 Macro keys
Automatic Capture of incoming files
X on/X off capabilities
Send True Line Break
Transmit/Receive BASIC Programs, Files or Machine Code. No need to translate BASIC programs to ASCII Format.
Save/Load Macros or Parameters to Disk
Use 1 to 4 Disk Drives (w/SAVE, LOAD, DIR & Granule Display)
Print while receiving information*
Easy to use MENU driven format
Comprehensive users manual
Works on All Radio Shack Color Computers, and All Radio Shack Disk systems.

15-day money back guarantee (less a $10.00 restocking/use charge.)
Only $5.00 each for all future upgrades when you return your warranty card.

PRICE: Ultra Term+ . . . . $59.95 (Disk)
  Y-Cable $29.95
  Double 80 Plus $99.95
  Complete Package . . . . $189.95 + S&H

*Requires PC Pak from PBJ, Inc.

DOUBLE 80 PLUS

TRUE 80 COLUMN OUTPUT
BUILT IN SWITCH FOR COCO OR DOUBLE 80 PLUS
ADJUSTABLE VIDEO OUTPUT
GOLD PLATED EDGE CONNECTOR
DRIVERS AVAILABLE FOR BASIC, OS9 and FLEX
DISPLAY ALL ASCII CHARACTERS
ALTERNATE CHARACTER SETS AVAILABLE
METAL CASE (not cheap plastic)
ULTRA TERM+ available for this board
BACKED BY A 90 DAY PARTS AND LABOR WARRANTY

DOUBLE 80 PLUS (80 column board) $99.95
Y-CABLE 29.95
BASIC DRIVER 12.95
OS9 DRIVER 12.95
FLEX DRIVER (available soon) 12.95
ULTRA TERM+ (disk only) 55.95

RECEIVED & CERTIFIED

The following products recently have been received by THE RAINBOW, examined by our magazine staff and approved for the *Rainbow Seal of Certification,* your assurance that we have seen the product and have ascertained that it is what it purpots to be.
This month the *Seal of Certification* has been issued to:

PC Index is an organizational tool consisting of a packet of 40 cards printed on both sides to fit on the protective envelope of your 5¼" diskettes allowing you to see at a glance the information a particular disk contains. ABACUS Computer Services & Education Group, P.O. Box 23438, Pleasant Hill, CA 94523, $2.95

Air Traffic Control Simulator, a 32K simulation designed to provide an orderly and safe flow of aircraft from their origins to their destinations following an established set of rules and procedures. You, as the controller, are given this set of rules as well as a choice of three levels of difficulty to develop techniques of skill, judgment and experience equal to performing the assigned task. Betasoft Systems, P.O. Box 1174, Smithtown, NY 11787, cassette $34.95, disk $37.95

The Designer, a 32K ECB Hi-Res program targeted for budding young fashion designers. Possible are the creation, saving and display of more than 1,000 style combinations in hundreds of color and fabric options. Cognitive Development Co., Suite 141A, 12345 Lake City Way, NE, Seattle, WA 98125, disk $24.95

Diet-Ade, a 32K program for health-conscious CoCo owners. A calorie counter is put at your fingertips, a calorie quoter tells you what to eat to attain your desired weight and a calorie minder nudges you to exercise to exceed your daily caloric limit. Color Connection Software, 1060 Buddlea Dr., Sandy, UT 84070, cassette or disk $19, plus $2 S/H

MAZERUNNER, a 16K ECB game requiring a joystick. As a member of the Maze Runner Corps, your mission is to fight off the forces of Ytirrod, the Pretender, from invading the Centrist Confederation and overtaking the Grand Dias from His Wisdom Arad Ennaed. You must destroy Ytirrod's fortresses by navigating remote controlled attack power sleds. Color Connection Software, 1060 Buddlea Dr., Sandy, UT 84070, cassette or disk $14.95, plus $2 S/H

Metabot, a 32K invitation to *le danse macabre* as you trick the menacing, mimicking Metabots into waltzing into fatal fences and lethal acid pots. Color Connection Software, 1060 Buddlea Dr., Sandy, UT 84070, cassette or disk $18.95, plus $2 S/H

CDSORT and **KEY-EXTR,** two 32K utilities requiring at least one disk drive. *CDSORT* allows the sorting of records of up to 256 bytes by more than one key field. *KEY-EXTRA* is designed to allow the extraction of key fields from databases and has the ability to include only specified keys in the extraction by means of 10 possible AND comparisons specified by the user. Computize, Inc., P.O. Box 207, Langhorne, PA 19047, disk $34.95

CONVERSIONS, a 16K ECB disk or cassette utility that makes short work of converting any unit of measurement to U.S., Imperial, or Metric. Choices include lengths, areas, volumes, liquids, weights and temperatures. Cozy Software, 25142-53 Ave., Aldergrove, British Columbia, Canada V0X 1A0, cassette $12 U.S., $15 CAN., disk $14 U.S., $17 CAN., plus $2 S/H

Disk Sort and Order, a 32K or greater utility requiring at least one disk drive. Compatible with Disk BASIC or *JDOS,* this program will sort directories alphabetically removing all "killed" filenames, change a Disk BASIC format of 35 tracks to *JDOS'* 40 tracks, restructure the entire diskette so that all granules for each file are sequenced saving wear and tear on the mechanical parts of your drive(s). Derby City Software, 3141 Doreen Way, Louisville, KY 40220, disk $11.95

SIDE WISE, a 32-64K program requiring disk drive that will read any ASCII text file from disk and print it out sideways down the page. Useful for obtaining printouts of spreadsheets that are too wide for the printer to handle. Fully menu-driven, *SIDE WISE* has selections of printer capabilities for Radio Shack, Epson, C-Itoh, Okidata and Gemini printers. Derringer Software, Inc., P.O. Box 5300, Florence, SC 29502, disk $19.95

Word Processor, a 16K ECB utility that includes ease-of-use commands such as; 'D' — displays whole file (except underlines) to screen as it will sent to printer; 'Z' —display of memory left and change of Baud rate, 'H' — help key displays list of commands. Drayon Software, P.O. Box 2516, Renton, WA 98056, cassette $5

ME-128-64, an upgrade kit to expand your 64K CoCo to 128K giving it a second bank which will run the same programs as the first bank. Complete instructions included. Dynamic Electronics, Inc., P.O. Box 896, Hartselle, AL 35640, $169

Multiprogram Manager, a 16-32K utility which when used with Dynamic's *96KX* allows the loading of five programs at once as a block. Dynamic Electronics, Inc., P.O. Box 896, Hartselle, AL 35640, cassette $14.95

Musx, disks one through five, a 32K monthly disk that plays eight to 10 current hits in four simultaneous voices with related graphics. GRAFX, P.O. Box 254, West Mifflin, PA 15122-0254, each disk $13.95 plus $2 S/H

Speed Racer, a 32K racing game requiring a joystick. The object is to pass the required number of cars for each lap. This number is indicated by an arrowhead under the "cars passed" marker. You must overtake these cars to progress to the next lap. Complete all five laps (10 miles) and you've won the race. MichTron, 576 S. Telegraph Rd., Pontiac, MI 48053, disk $34.95 cassette $34.95

Bakersfield KENO, a 32K ECB game of chance adapted from the game Lotto. The player chooses an initial bankroll and one to 15 numbers to wager on. The payoff on winners increases as more

numbers are played and hit. It's possible to win 1,000,000 "dollars" with daring and prowess. SEIBYTE Software, P.O. Box 6464, Bakersfield, CA 93386

Chipaway, a 16K ML game requiring one joystick that pits your skill with a paddle and ball against a wall of multi-hued chips. Each color chip gives a different point value when hit. Options include FLIP, which flips the wall over thus displaying high point chips; COLLAPSE, which causes any chips above the chip that the ball hits to collapse down a notch and a game save routine. Sigma Software, 14024 152nd Avenue, S.E., Renton, WA 98056-7313, cassette $5.95

LINER, a 16K ECB game that casts you in the role of small game hunter. Your quarry: the tiny Eeblewalkers of Oktry who try and turn your system of locomotion against you. Michael Stuller, 2 Audubon Place, Rolla, MO 65401, cassette $7.95 plus $1.50 S/H

SBASIC, a 32-64K pre-compiler that adds commands and constructs to the CoCo. Among commands/constructs included are: *LOOP/UNTIL(), IF/THEN/ELSE/ENDIF, Continue, Quit* and 64K version allows source and object code in memory at once (save and retrieve with one keystroke). Tandar Software, 12 Araman Drive, Agincourt, Ontario, Canada M1T 2P6 specify disk or cassette $19.95 U.S., $24.95 Canada, $3 S/H U.S., $1 S/H Ontario and Montreal

MLBASIC, a 64K enhanced BASIC compiler designed to allow greater compatibility with existing Interpreter BASIC programs. A full compiler that features all of the commands available with Standard, Extended and Disk BASIC. *MLBASIC* offers additional commands to make it possible to interface programs with assembly language and write structured programs. WASATCHWARE, 7350 Nutree Dr., Salt Lake City, UT 84121 cassette $69.95, disk $69.95, disk $69.95 both $74.95, $4 S/H

The *Seal of Certification* program is open to all manufacturers of products for the TRS-80 Color Computer, the TDP-100, or the Dragon-32, regardless of whether they advertise in THE RAINBOW. By awarding a *Seal,* the magazine certifies the program *does exist,* but this *does not* constitute any guarantee of satisfaction. As soon as possible, these hardware or software items will be forwarded to THE RAINBOW'S reviewers for evaluation.

— **Monica Dorth**

SUPER DISK UTILITY

Editor:

In regard to the review of *Super Disk Utility* I am returning a few facts that I feel need to be stated or clarified in the review.

It appears that the review is single sided towards the cataloging function and is not proportionally written with the functions supported by *Super Disk Utility*, and certain functions were not reviewed on their ability to do the task advertised or stated in the manual.

The reviewer states that there is not a search or find function in the catalog. When the program was originally written it was the author's choice not to include this function since a print function is provided to print the catalog in a neat, readable format. However, it should be stated that an updated version is being created which includes a search function and the following functions: Delete information in catalog, Add information to catalog. The fact that the same catalog can be sorted and printed in three different ways was neglected in the review of the catalog function.

The *Diskzap* function of the program does not review its functions; instead it merely lists them and includes a commentary on the display options (which I feel could have included a printout of the displayed disk sector and a command summary of the modification mode included on pages six and seven of the *Super Disk Utility* manual). An important feature of the *Diskzap* is the allocation table check and repair. If a disk develops a file structure error or an end of file error this function can be used to fix it and save some of the files on the disk. An example of testing this feature would be to back up a disk and modify track 17, sector 2 and change some of the first 68 bytes, (change a CO-C9 to anything other than CO-C9). Whether the *DIR* command in BASIC will find an error or not, the allocation table check will. Use this function and follow

instructions on Page 8 of the manual.

It should also be noted that the sector skip factor is not a feature of just non-Radio Shack drives but is, in fact, a software controllable factor.

It should also be clarified that the *Diskzap 40* will work on standard Radio Shack disk drives, but it will not access above 37 tracks, and modifications to the program are allowed by the user for this purpose. The save directory feature is an important one to people who value their software—explanation: The Radio Shack DOS system is set up so that the drive head is on track 17 most of the time — reading and writing is done here every time a file is loaded or saved and failures in the drive can and do occur; therefore I created the save directory function to protect the disk from BASIC because BASIC does not access tracks 36 and 37.

I also feel that the section of the review on purge files, copy files, date files and directory sort could have gone more in depth on these functions. The reviewer neglected to mention the Super Directory function which uses the date function to get its dates and also computes the start, end and executing address of machine language programs. Each file is printed to the screen or to the screen and printer in a neat, easy to read format.

Bryan Woodruff
Petrocci Freelance Associates

Editor:

In response to Mr. Woodruff's comments on *Super Disk Utility*, I offer the following thoughts:

Mr. Woodruff's feelings about the "balance" of the review may well be just a difference of opinion between the supplier and consumer (albeit surrogate). I freely admit that I did not spend much time discussing some of the "bells and whistles" in the offering. Some of the items do things that are

already built into the Color Computer while some others are the same as programs which have appeared in THE RAINBOW.

I am pleased to see that Mr. Woodruff is planning a revision to include a "search" function. This will add greatly to the utility value of the program.

Lastly, I agree with Mr. Woodruff that the "sector skip factor" is controllable on Radio Shack equipment but I felt that this would be of more interest to non-Radio Shack drive users since they might make more use of the function.

Robert E. Foiles
Lancaster, PA

MORSE CODE
TEACHER AND TUTOR

Editor:

As both a reviewer for THE RAINBOW magazine and a CoCo software author, I couldn't help but notice Cynwyn's rebuttal letter (Reviewing Reviews Sept. '84) concerning the review of some ham radio software that was reviewed by someone who doesn't know much about the subject.

I have always been happy with the programs you send me to review because they fall into my lines of interest and experience. It shows great care on your part in most cases to line up programs with the reviewer.

I don't like to criticize the work of another reviewer but in this case I will. At the time that Cynwyn's *Morse Code Teacher and Tutor* was reviewed (June '84, Page 250) I was in the need for a program just like that. The review did not give me any of the important details needed to make a wise purchase so I didn't order it. Just recently I saw the program demonstrated and it was quite good, much better than the review.

For a small company like Cynwyn this review may have cost them that shot in the

arm needed to succeed, even if in a small way, in the CoCo software world.

Here in the Bay Area, CoCo users put a lot of stock in THE RAINBOW reviews, and that's one of the reasons I try to do thorough reviews.

J. Michael Nowicki
San Jose, CA

STYLO III FOR OS-9

Editor:

In reading Dale L. Puckett's review, "Stylo III For OS-9 — What You See Is What You Get," Oct. '84, Page 192, I was amazed to read ". . . it shows them exactly what they are going to see on the printed page on the screen while they are typing it. *Stylo* is the only word processor that brings this feature to the Color Computer. Others feature screen editing, but they do the final text processing while they print."

This is a gross inaccuracy. *Telewriter-64* does just this — better known as dynamic formatting.

A. David Muir
British Columbia

PICTURE PUZZLE

Editor:

There seem to be some misconceptions concerning the program *Picture Puzzle*. First

of all, nowhere in the documentation does it state that the user can create his own pictures using this program. To create pictures, we highly recommend a picture-designing program such as *Graphicom*. I believe the option the reviewer is referring to is the "Add Option." This will only allow adding a binary picture file into the picture directory.

Next, he refers to the poor quality of speech. We feel it is the finest speech available using single bit sound without any hardware. Poor quality could be the result of a poor speaker system.

We did try to return the reviewer's call on two occasions. I am sorry, but this matter did slip our mind. We make a conscientious effort to return every phone call and provide assistance when needed.

If you have any further questions, please do not hesitate to contact us.

Dennis S. Lewandowski
DSL Computer Products

Editor:

I agree with DSL's remark that their documentation does not mention how to create a new puzzle, however, let me explain why I found fault with this part of the program. The menu specifies, as choice number one, "Add New Boards." When this is chosen the screen shows, "Reading Data From Disk" and lists the three programs that come with *Picture Puzzle*. A question is then asked, "How many new ones?" You are required to type a number and the program then asks for the "filename". Thus, the menu option, plus

these specific questions certainly implies that a picture can be added to the program by the user. In addition, the documentation stated that, in adding a picture, you should not use the same design in two squares of the puzzle. The documentation not only does not state that it is necessary to buy an additional graphics designing program in order to add a picture; it does not explain how to add a picture even if you do have one.

While DSL may feel that the "talking" of *Picture Puzzle* is best without using hardware, my opinion is that the result is of low quality voice reproduction. As I said in my review, it is a good attempt but far from perfect. Please realize that I feel the program itself is quite enjoyable and while the talking does not add to it, it does not detract either.

I am glad DSL tried to return my call. As a mail order customer who paid for a program and then made a long distance call to clarify the documentation, I strongly object to not getting a reply when I am told I will be getting one. Perhaps DSL, and other companies, could request a customer's telephone number and address. If the company were unsuccessful in returning a telephone call they could send a form post card stating, "Sorry! We tried to contact you but could not. Please either write us with your questions or call again."

Michael F. Garozzo
Morrisville, PA

CoCo Checker
Best In Its Class

I first got involved with computers through my interest in electronics, and I got started in *that* by fixing TV sets. One of the beginners' books I read pointed out that the TV set would almost tell you what the problem was, if only you knew how to interpret the picture you saw. Computers are like that in some respects; as an example, if you find that you aren't getting any signal on your cassettes when you save a program, you can figure that the single-bit sound circuitry isn't putting out as it should be. A computer, being the versatile device that it is, can even help track down its own troubles, assuming enough of the machine is still working.

Radio Shack has been selling its *Diagnostics* cartridge ever since the CoCo came out in 1980 (it's still available by special order), but it has some limitations and tests only the computer itself. Spectrum Projects' *CoCo Checker* not only gives the CoCo a workout, but exercises the disk controller, disk drives, printer and Multi-Pak Interface, if you have them.

CoCo Checker comes on either cassette or diskette; I tested the disk version. Considering the nature of the program, a ROM pack might be more likely to be usable in some cases, but would be more expensive (and would make it harder to test the disk system). Bob Rosen at Spectrum suggested that if you have a disk system, you could copy the program from disk to cassette when you get it, so you aren't dependent on loading it by one means alone later on.

As one might expect, *CoCo Checker* has test routines for the cassette port, joystick ports, keyboard, serial I/O port, RAM, ROMs, sound circuitry and video generator; all of these are quite complete. The only RS-232 testing is part of the printer test, so you need a printer to check your machine's serial I/O circuits. (Radio Shack's cartridge uses a specially wired shorting plug, which they don't supply, to feed the CoCo's data output back into the two input lines.)

This test also gives a more complete test of the printer than the "self-test" mode it probably has, since it checks the entire signal path. The ROM checksum test is set up for all the ROMs Radio Shack has released so far. The keyboard test even checks the function keys found on the HJL, Key Tronic and Macrotron keyboards. There is even a test for timing accuracy, comparing the horizontal and vertical blanking interrupts to the master clock.

The unique features of *CoCo Checker* are the disk controller, disk drive and Multi-Pak tests. The controller test runs the controller IC through various steps and checks the interrupt line. The drive test checks drive speed and then writes a test pattern to the diskette in the designated drive and checks for errors. (This of course means a bad disk can cause a good drive to appear bad.) The Multi-Pak Interface test routine gives the unit's PAL (programmable array logic) chip a workout and also checks the select switch circuit.

If you do any repair work on CoCos at all, this program is an absolute must. Even an ordinary user can benefit from *CoCo Checker*, as it tells you whether or not the system is working properly and even what to tell the repair technician when it isn't. Now if only there was a video cassette that could do this sort of thing on TV sets!

(Spectrum Projects, Inc., Box 21272, Woodhaven, NY 11421, or Box 9866, San Jose, CA 95157-0866, $19.95 plus $3 S/H)

— Ray Edwards

Submitting Material To The Rainbow

Contributions to THE RAINBOW are welcome from everyone. We like to run a variety of programs which will be useful/helpful/fun for other CoCo owners.
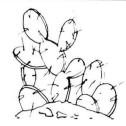
Program submissions must be on tape or disk and it is best to make several saves, at least one of them in ASCII format. We're sorry, but we do not have time to key in programs. All programs should be supported by some editorial commentary, explaining how the program works. We're much more interested in how your submission works and runs than how you developed it. Programs should be learning experiences.

We do pay for submissions, based on a number of criteria. Those wishing remuneration should *so state* when making submissions.

For the benefit of those who wish more detailed information on making submissions, please send a SASE to: Submissions Editor, THE RAINBOW, P.O. Box 385, Prospect, KY 40059. We will send you some more comprehensive guidelines.

Please do not submit programs or articles currently submitted to another publication.

Unscramble Those Boxes With *High Resolution Picture Puzzle*

Remember those little puzzles with eight sliding numbers and one blank? They were numbered one to eight and after you messed up the order you had to slide each number around until you could get it numerically correct. The 3 x 3 box was not too difficult to solve. So how would you like 48 boxes to unscramble, not only with numbers but with two different types of pictures?

Picture Puzzle is perfect for those of you who like to unscramble puzzles without having the extreme of a Rubik's Cube. In addition, the disk version "talks." I feel that the "talking" is similar to listening to a person using a poor quality loud speaker system. It is a good attempt at producing software speech but is far from perfect.

The puzzles are very well done. In addition to the numerical puzzle (one to 48), there are two picture puzzles. These are the "hat shape" (this design is featured in DSL's advertisement), and an optical illusion design triangle. After the disk version says, "CAN YOU SOLVE PORKKA'S (author) PUZZLE? PLEASE CHOOSE ONE OF THE BOARDS FROM MENU," the menu appears and you can choose which puzzle to unscramble. The program shows you the completed form of the puzzle and then clears the screen and

mixes up the puzzle by scrambling the 48 squares. You then have five minutes to unscramble the puzzle by moving the squares. Each square is moved into the one blank space by using the four arrow keys. By manipulating the blank space you can reassemble the puzzle. The author realized that it would be necessary to view the completed puzzle from time to time, so there is a help key, 'H', that pauses the timer and allows you to see the completed puzzle. When you are out of time, the disk turns on and announces, "TOO BAD, YOU'VE RUN OUT OF TIME. JUST A MOMENT. LET ME SEE IF YOUR SCORE IS IN THE TOP TEN." If your score is a high score the program will ask your name and save your score on the disk. You can continue with another puzzle or repeat what you have just done. If you solve the puzzle the statement, "CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THIS ROUND," appears and the computer then allows you to play again. But each time you solve the puzzle the computer scrambles the puzzle more than it did previously. This is truly a never-ending game.

There is a bonus section on the disk that states that you can create your own puzzle pictures. The instructions say not to have two areas of the puzzle exactly the same or the puzzle will look solved but may have the two blocks swapped. This will cause the computer to lock up.

This is the section I had problems with. The directions are limited and do not explain how to make up or design a puzzle for the program. On the menu selection of "add new boards," the only thing that happens is that the program writes a new title to the disk. This new title will appear on the menu but there is no way to develop a new design. In order to solve this problem I telephoned DSL Computer Products. Their office was closed on the day I called (Monday) but a tape recording said that if I left a message they would get back to me. I explained the problem and left my phone number. They did not return the call. I do not appreciate getting a tape recording telling me an office is closed and I despise being told to leave a message if no return call is planned. I should mention that I waited eight days for a response. Many computer users rely upon mail order companies and those companies should be able to work with and help purchasers of their programs.

Should you purchase *High Resolution Picture Puzzle*? Well, do not buy it for the "talking," and do not buy it for designing your own puzzles, but buy it because the author, Joe Porkka, has done an excellent job of giving you new puzzles to solve and hours of fun.

(DSL Computer Products Inc., P.O. Box 1176, Dearborn, MI 48121, 32K Ext. BASIC, tape $9.95, disk $12.95)

— Michael F. Garozzo

Option 1 Is Number One

Are you tired of digging through that old small appliance box trying to find a canceled check, only to find it in the very bottom? Or devoting an entire wall of filing cabinets just to store all those checks? If you are like me, you always seem to lose the checks that you might need later. Maybe you have one of those little critters that eat some of your canceled checks (I can never figure out how these little critters seem to know which one to eat).

We finally have a Cash Disbursement Program that will stack up against the big boys called *Option 1* by YGS Software.

After loading *Option 1* you are greeted with the maple leaf from our beautiful sister country, Canada, then you are prompted to enter your own unique password (if the wrong password is entered you will be presented with an illegal usage message and forced to shut down all equipment in order to regain control of your system).

Next will be the main menu consisting of:

1) Issue Checks
2) Record deposits
3) Transfer tape file to disk
4) Sort/View by payee or account number
5) View records by check
6) Obtain balance excl. pay
7) Change records

After your selection has been made, each routine will be loaded into the system individually. When issuing checks or recording deposits a check will appear on the screen and prompt the user to input all of the correct data. The printed checks feature detailed stub printing, account number, reference number, dollar amount, check number, discounts, payee, address, city, state, ZIP code and memo.

Option 1 is designed to print on formatted checks available through New England Business Systems, however if you have your own pre-printed checks you may have to alter the printer driver to accommodate your own needs. Whichever the case, *Option 1* prints crisp, clean checks at the touch of a button.

The program checks every keystroke making sure the correct data is keyed in the proper place, resulting in the process being somewhat slow, but on the other hand it could be a timesaver by keeping the user from keying in an entire check all over again. After practice the process could become considerably faster.

Option 1 is machine language and consists of a printer driver for the Epson MX-80 printer or compatible. If you do not have a compatible printer you can write your own printer routine, or you can send your printer's manual to YGS and they will write a routine for your printer at an hourly rate. Also if you ever want your password changed you can send the original program to YGS and they will change your password for 15 percent of the current retail price of the entire package.

The disk can be backed up but is not executable; first you must back up the disk with normal backup procedures and place your new copy someplace out of the way. If your original disk ever becomes inoperable you can copy the backup back onto the original disk (if the original disk is damaged the backup procedure will not work).

A few suggestions that I feel should be in business software are:

Two disks should be included in the entire package, eliminating down time as a result of sending the original disk in for a password change.

Option 1 is written for a one-drive system. An option should be included using a two-drive system, eliminating the need to switch between the main disk and a data disk. (There is a lot of disk switching involved.)

If the user forgets to take out the main disk and place the data disk in the drive and tries to save any data onto the disk without removing any write protect device, the system will crash, resulting in a loss of data that was just input from the keyboard. The system must be reset and reloaded in order to regain control. There needs to be an error trapping routine in this section to eliminate the would-be loss of what could be valuable information.

Upon receiving my copy of *Option 1*, I was blessed with a bad disk copy. After fighting with the disk trying to get it to load, I made a phone call to YGS Software. Assuring me that they knew what the problem was and it had been corrected, they sent me a new copy promptly. My experience with YGS was both courteous and a pleasure as they handled my problem tactfully and swiftly. So throw away that old small appliance box and devote all that wasted wall space to your CoCo and, above all, run that little critter that eats all those important checks out of business . . . I did.

(YGS Software, P.O. Box 208, Brechin, Ontario, Canada L0K 1B0, $56 U.S., $69.95 Canada)

— Bob Brown

Tic-Tac Math
Improves Arithmetic Skills

Tic-Tac Math is a tic-tac-toe game in Hi-Res graphics combined with exercises in solving math problems (addition, subtraction, or multiplication, *no* division) at a level of difficulty chosen by the student at the start of each game.

Tic-Tac Math is easy to *CLOAD* and *RUN*. All other instructions and an explanation of the game can be displayed while using the program. After the title page you are offered a choice of three types of problems: addition, subtraction and multiplication. You must then select the level of difficulty (easy, medium or hard). The level of difficulty in *Tic-Tac Math* is *not* increasingly more difficult problems such as two- or three-digit numbers, but rather a decreasing amount of time available to answer the problems! Easy gives you three seconds, Medium — two seconds, Hard — one second. All of these times can be changed by passing through the instruction screen, which comes up next after selection of difficulty. If you choose not to see the instruction screen you go directly to the gameboard screen, and the timer remains at one, two or three seconds.

The instruction screen gives you brief instructions, an explanation of the game and the opportunity to either increase or decrease the response time allowed for each problem. The documentation (one typewritten sheet) gives a conflicting explanation of this response, but the program's screen was correct: Pressing the 'U' speeds up the response time (gives you less time to answer) and pressing 'D' slows down the time (gives you more time). For each time the 'U' or 'D' is pushed, the time is decreased or increased by a factor of 1.5, e.g., the Easy level originally gives you three seconds. Pushing 'D' once gives you 3 x 1.5 or 4.5 seconds. Pushing 'D' again will result in 4.5 x 1.5 or 6.75 seconds. Pushing ENTER on the instruction screen sends you to the gameboard screen.

The gameboard is drawn and divided into numbered squares (one through nine). You are prompted to choose a number and are in full control of which squares are to be played. You do not have to play a square just to block the computer. Immediately after choosing a number, a problem appears on the screen to the right of the gameboard and the timer begins. If the problem is answered correctly *and within* the time limit, the student wins the square with an 'X'. Each correctly answered problem gets one of three responses placed on the screen: "CORRECT," "GREAT," or "TERRIFIC." If the time runs out or the answer is incorrect, then the computer wins the square with an 'O' and "INCORRECT" appears on the screen.

If neither you nor the computer wins the game by the time all the squares are played, the board is rebuilt and play begins again using the options selected for the previous game. If you are successful and complete a tic-tac-toe, you are rewarded with ascending notes and "YOU WON" is placed on the screen below the board. If you lose, "COMPUTER WON" appears and descending notes are heard. You then go back to the selection of the type of problem and start the cycle over again. If you do not pass through the instruction screen, the timer is set according to the difficulty selection and does not keep the setting of the previous game. This bothered my kids somewhat at first.

A couple of things bothered me about *Tic-Tac Math*. First and foremost is the level of difficulty. The differences between difficulty levels using a time factor are somewhat insignificant as the student can pick the Hard level and transform it into a Super Easy level through the instruction screen, as my son, age seven, usually did. The unmodified Hard level is very discouraging even for an adult math major with touch typing. Mentally one knows the answer the instant the problem is placed on the screen. However, finding and pressing the right keys in less than one second is very difficult.

I much prefer the option of working with more difficult numbers. This was also expressed by my oldest daughter (age 10). The actual range of numbers used are one-digit numbers from two to nine. There are no ones or zeros. The student is given no second chance at a wrong answer and more importantly, I feel, there is no correct answer shown if your answer is wrong.

Nevertheless, my son still enjoys and talks about *Tic-Tac Math* even after several days of playing. *Tic-Tac Math* can provide the slow learner in math with ample time to solve problems and be rewarded with the accomplishment of getting tic-tac-toe. And it can provide competitive excitement to those early grade school students who are more masterful of mathematics. If your children are in the early developmental stage of math, *Tic-Tac Math* can provide a good foundation for basic math while they enjoy playing a game of tic-tac-toe.

(Wish Software, P.O. Box 7366, Johnston, RI 02911, 16K ECB, tape $24.95)

— **Kenneth D. Peters**

DATAMAN

INTERNATIONAL

VISA   1-416-529-1319   MC

DEALERS

YOU CAN RECEIVE A 30 MINUTE IN STORE
DEMO DISK THAT IS GUARANTEED TO
❖❖❖❖ *BLOW YOU AWAY* ❖❖❖❖
THE GRAPHIC/TEXT MASTER DEMO DISK
CONTAINS BOTH M/L PROGRAMS PLUS A
BASIC DEMONSTATION OF THE INCREDIBLE
VERSITILITY OF THESE NEW   "STATE OF
THE ART" UTILITIES.

SEND YOUR REQUESTS ON COMPANY LETTERHEAD ALONG WITH $20.
WE WILL SEND YOU THE DEMO PACKAGE AND A CREDIT FOR $20
WHICH YOU CAN APPLY TO YOUR NEXT ORDER.

DATAMAN
INTERNATIONAL
125 SOUTH FIFTH STREET
LEWISTON, N.Y. 14092

DATAMAN
INTERNATIONAL
420 FERGUSON AVE. N.
HAMILTON, ONT., L8L 4Y9

Play The Lottery
With *Lotto*

With taxpayers across the country jerking their collective knee at the mere mention of raising taxes, some of our state governments have initiated lotteries that give everyone an equal chance at becoming a millionaire and at the same time lotteries bring in some much needed revenue to the state coffers.

Like most other gamblers, lottery players have their own techniques of selecting the winning numbers to write on their ticket. Some use special dart boards, patent numbers on manhole covers or intricate formulas like multiplying their birth date by their social security number and then dividing that figure by the total albatross population of Tasmania. It's no wonder that in this high-tech world someone would write a program to handle the number selection process for you.

This program, called *Lotto,* requires 16K cassette and is written in BASIC. All you do to get started is to *CLOAD Lotto* and *RUN.* The instructions scroll across the screen at a very slow rate and it takes about two minutes for them to complete. No hard copy documentation is included, or for that matter, is needed.

Lotto is written using the state lottery rules as found in Maryland, but instructions are given for making the simple changes to match the rules in any state lottery. Unfortunately the line numbers referenced for change are not accurate or complete and some knowledge of BASIC by the user will be required to spot the error and make the right changes. A spelling error was noticed on one of the screen prompts but that can be easily fixed as well.
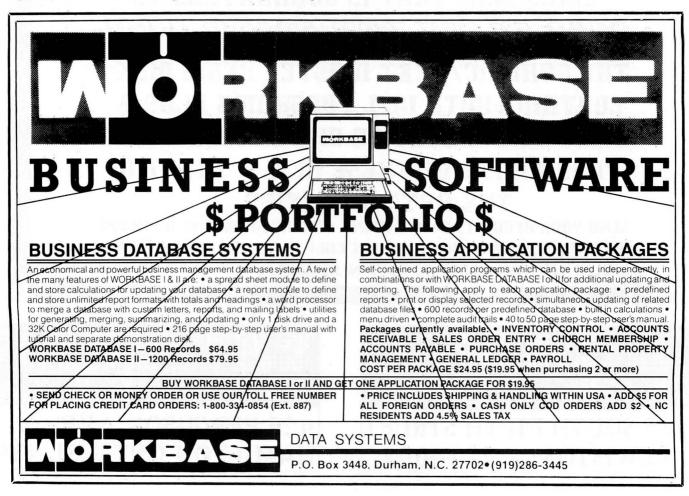
What does this program do? It picks out 40 or more different random numbers and displays them on the screen for you. It uses BASIC's *RUN* command to get the numbers but as many of you know, there is a bug in the *RUN* command that can be somewhat cured by including a program line that includes the statement $X=RND(-TIMER)$. This fix is not included in the program so the random numbers generated are not quite as random as they could be.

During the number selection process the text screen is flashing a different color for each number generated and then all 40 numbers are displayed in no particular format or grouping.

Even if there were no bugs or errors in the program, the low purchase price does not justify plunking down any amount of cash for such a simple program that even a novice programmer could hack out in about five minutes.

(RAM Publications, 1088 Poplar Tree Drive, Annapolis, MD 21401, tape $5.95)

— J. Michael Nowicki

The Animator Turns CoCo Into An Animation Studio

A lot of attention has been focused lately on computer-generated animation used in popular movies like *Star Wars,* commercials and even demonstration films like *Panasonic Plane* (where the company's 3-D TV system is shown off by a paper airplane flying around a house). Next to this sort of thing (which even local TV stations are using for news and weather coverage these days), what you typically see on your home computer looks rather tame. Triad Pictures' *The Animator* can give you a taste of computer-generated animation on the CoCo, and teaches you about the way studios like Warner and Disney do animated cartoons.

The program's sections are arranged (and named) like the departments of a film studio. The "background studio" is where you draw the backgrounds for the scenes. The "cel animation studio" lets you draw the various objects (as if on the cellophane sheets used for film animation), and the "camera department" combines backgrounds and cels to create the finished picture. The "sound effects studio" lets you add sound to the picture, and the "screening room" is where you watch the finished presentation. Finally, the "lab" makes copies of the film or of the background and cels.

The background section is more or less a normal Hi-Res screen drawing program, working only in *PMODE 3.* You can move and draw in all directions using the arrow keys, change colors and paint enclosed areas. The cel animation section is similar, only here you are drawing as many as 32 small objects (or variations on an object) instead of a whole scene. The cel printer function lets you "take a picture" of a particular cel and "print" it in one or more cel locations, so you can repeat all but a small part of an object.

In the camera department you combine the background and the objects on the cels to form frames. In each frame you can place one or more of the cels over the background in desired positions (or have a frame with no objects), then shoot a frame and go on to the next. When you have shot all the frames in your sequence, you cut the end of the film and return to the main menu. There is also what the author calls a "rotoscope;" it's similar in function to the 8mm home movie editors that used to be common (or more like a good industrial video recorder), in that it lets you look at the film step-by-step or at normal speed in either direction. You can even put the "film" back in the "camera" at any desired frame (sounds more and more like video, doesn't it?) to shoot over from a desired point. You can shoot up to 625 frames.

The sound effects section has 12 commonly used sounds in its library. To use a sound effect you simply use the "rotoscope" viewer to locate the point at which you want it to come up, then select the desired effect and press 'I' for insert. Finally, the screening room "projects" the entire

cartoon with sound — and does it over and over without stopping until you press 'D' to go back to the main menu. The lab saves or loads a background, a set of cels, both of them or an entire presentation (consisting of background, cels and instruction codes to re-create the finished product).

All the basic functions are explained in the first half of the manual; the second half is a 16-page animation textbook written by the program's author, Dwayne Osterbauer. In it he talks about advanced animation techniques in great detail; he speaks mostly in "film" terms (I get the feeling that he's either a very avid cartoon buff or has been/is in the business himself), but fully explains how they fit on the CoCo using his program. Several demonstrations are provided that go with the book.

The one thing that, more than anything else, causes the obvious difference between *The Animator* and real animation is not the relatively low resolution (it doesn't seem so bad), but the low frame rate, roughly seven or eight frames per second. Motion seems much jerkier than even the low-budget Hanna-Barbera cartoons, which were shot at only 12 frames per second. The frame rate is understandable, given the juggling act involved in moving cels (the program has to redraw the cels over the background for each frame and add sound, all in real time).

The Animator's graphics can, of course, be recorded in one of several ways. If you have a video recorder, just feed the CoCo's RF output into the VCR's VHF antenna input, set the recorder tuner to the right channel and away you go. Unfortunately, it's very difficult to stop the tape at exactly the right point for the end of a scene and restart for the beginning of the next scene (if that's your intention)

on home-type recorders. You could also use an "XL type" Super 8 camera on a tripod to shoot from the TV set (older movie cameras don't work well in this application); editing is more precise here, and if the camera has a single-frame option you can place one frame of animation on each frame of film, bringing the frame rate up to 18.

The Animator certainly is a one-of-a-kind package; I've never seen something like this on any other home computer. This seems like a very good buy for the CoCo.

(Triad Pictures Corp., Box 1299, Sequim, WA 98382 cassette $35)

— Ed Ellers

One-Liner Contest Winner . . .
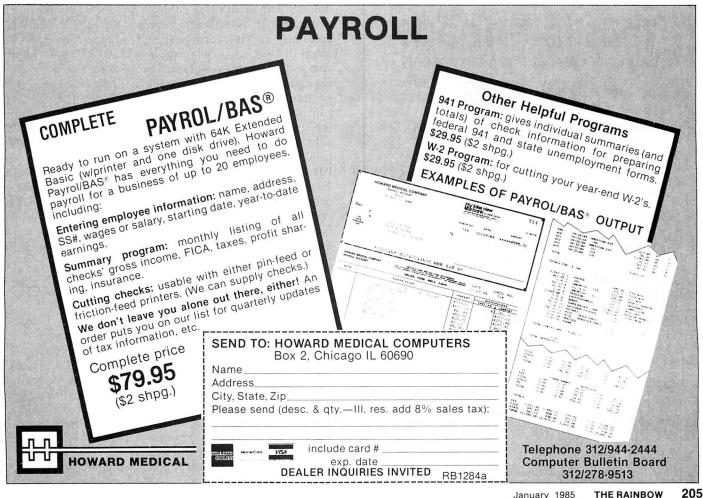
For those of you who like nice, neat printouts of disk directories, this will do the job nicely. Simply put your printer online, insert the desired disk in drive 0, ENTER its name and the program does the rest.

Dave Helfrich
New Port Richey, FL

The listing:

```
10 PRINT#-2:INPUT"DISK NAME";X$:
PRINT#-2,X$:PRINT#-2,"--------
----------":POKE111,254:DIR:PRINT
#-2:PRINT#-2,"FREE SPACE"FREE(0)
"GRANULES":PRINT#-2:PRINT#-2,"--
----------------":GOTO10
```

(For this winning one-liner contest entry, the author has been sent copies of both *The Rainbow Book Of Adventure* and its companion *Rainbow Adventure Tape*.)

Organize Your Disk Library With *Where's-It*

By A. Buddy Hogan

Have you ever asked yourself, "Now where is that program that lets me copy files from one disk to another without doing all that typing?" Or some similar question that has to do with a lapse of memory concerning the location of a program on a disk? Everyone who has a disk collection of more than a few disks will ask such a question sooner or later. *Where's-It* is a utility that allows you to answer such questions. It is a straightforward program that does what it is advertised to do.

Where's-It is simply a disk index database management utility program. It allows you to create a database consisting of the directories of as many of your disks as you desire (however, no more than 972 program names can be used to build each index). You are then able to manipulate the data to suit your needs. *Where's-It* is supplied on disk and is compatible with the 32K Extended BASIC Color Computer and requires at least one disk drive. The main program is written in BASIC while the sort routine is provided as a separate machine language program on the disk.

Upon running *Where's-It*, you are presented with a well designed menu screen that displays the nine options availa-

ble to you. I'm going to list these options as they would be used rather than in the order presented in the menu (software developers would do well to present menus in this manner so that they are more useful):

1) ADD NEW DISK — This is the option that allows you to input the directories from your disks. The procedure is very simple. You are asked for a disk identifier (not more than five characters long) and then are directed to place the disk in the proper drive (the documentation tells you how to modify the program for two-drive operation). The directory is then read in very fast and you are asked for another disk identifier so that you can quickly add as many disk directories as you desire. I used 13 full disks containing 340 programs to test *Where's-It* and was able to build the database in less than seven minutes. The program actually stores the program name, the extension and the disk identifier for each program on the disk being added to your index.

2) SORT — allows you to sort the contents of your database in alphabetical order without regard to upper- or lowercase. The sort is fast since it is done in machine language. My 340 program index took less than a minute. An audible tone lets you know when the sort is complete. A nice feature would be the option to also sort by disk identifier. This way you would be able to see at a glance all of the files on a particular disk. You would obviously have to be able to store the identifier sort as a separate file.

3) LIST INDEX — prints the index file to the screen 10 programs at a time. ENTER advances the screen display to the next ten programs. An up-arrow returns you to the menu.

4) PRINT INDEX — prints your index file on an 80-column printer in three columns with a maximum of 162 program names on each page (54 in each of the three columns). This is where the 972 record limit comes from (six pages of 162 each). The printout is clearly the most impressive feature of this program. The print routine was written for an Epson FX-80 and provides a title for each page of the printout that displays DISK DIRECTORY INDEX at the top in expanded print. The next line, still in expanded print, is FILENAME: XXXXXXXX - PAGE X where the X's are substituted with the proper filename and page number. The printout continues with three columns, each headed with FILENAME EXT DISK. These headings are in the normal Epson print font and are underlined. The print routine then prints out the filename, extension and disk identifier for each program that is contained in your index. After 60 lines, it skips to the top of the next page, places the printout title lines and column headers at the top and continues the printout.

5) SEARCH — lets you find either a program by name or a particular disk by the identifier that you have assigned to it.

6) DELETE/UPDATE DISK — you may delete the directory of a disk from your index or add the directory of a disk that you have updated since it was first entered into your index by selecting either DELETE or UPDATE. You are then prompted for the disk identifier of the proper disk.

7) SAVE FILE — your index file is saved to disk with this option. If you started the current session by loading an index or have saved the current index once before during the current session, you may save the index by simply ENTERing up-arrow after selecting this option from the menu.

8) LOAD FILE — is used to load a previously saved index from disk.

9) MAKE NEW FILE — erases any data in memory and allows you to start building a new index. Since this is a destructive feature, the program prompts you to be sure that you want to use this option.

At the bottom of the menu display is the filename of the index that was loaded, the number of programs contained in the index and the number of programs that can be added to it.

The program prompts you for input by flashing the cursor at the proper location. Input is taken immediately without the need to hit ENTER except for a filename or identifier. Error handling is excellent. If you have a problem while using the program, just ENTER BREAK and then *GOTO 100* to get back to the main menu. You are not allowed to select options that require an index to be in memory if you don't have one there and you are not allowed to SORT or SEARCH if you only have one record in memory.

The documentation consists of five 8½ x 11-inch sheets of yellow paper printed in the normal fashion and folded in the middle. The documentation is adequate but could be improved. It leaves one wondering just how you are to begin to use the program. If you follow the sequence of the options as presented, you begin with MAKE NEW FILE (which talks about erasing the contents of memory — before you have entered anything into memory) then go to *LOAD* (you have nothing to load yet) and *SAVE* (save what?) before you get to ADD NEW DISK (which is where you begin).

The most glaring omission from the documentation is in the part that describes the *PRINT INDEX* option. You are told that the print routine was written for an Epson FX-80 and you are told that you might need to change some of the print variables (these are identified along with the line number where they need to be changed). However, when you get to the section on the Baud rate, you are informed that the Baud rate for the FX-80 is 9600 and that, "You may have to delete this line or replace it with some other value." End of instructions. It would not have been that difficult for the distributor to have informed purchasers of the program of the values needed to be used in Line 801 for the various Baud rates that different printers require. My Epson MX-80 worked just fine after Line 801 was deleted. Running it with Line 801 produced garbage on the printer since the Baud rate was much faster than my MX-80's 600 rate. Perhaps this observation comes under the heading of nitpicking since the owners of printers that run at other than the standard CoCo 600 should already be aware of the value that they need to poke into memory location 150 to produce the required Baud rate.

Where's-It is a worthwhile program that performs as advertised and would be a welcome addition to the collection of CoCo disk users who have accumulated enough programs on disk that they have to ask themselves that troubling question from time to time, "Where's it?".

(Jarb Software, 1636 D Ave., Suite C, National City, CA 92050, $19.95)

CoCo Can Engineer
With *Space Frame*

Because I used overlays years ago to "shoehorn" a major rotary-winged aeroelastic analysis into an early mainframe, I *know* 16K is big enough for serious work! Because I have edited reports for publication, I *know* a word processor is not a glorified electric typewriter. A spreadsheet is as far ahead of a hand calculator as the calculator is ahead of a slide rule.

But when the disparaging question is posed, "Can you really do anything *professional* on that little machine?", a negative answer can result from a limitation in the software market. I do not expect to find a supersonic wind tunnel calibration program in this magazine, and if I want another aeroelastic program (Color Computer compatible), I had better write it myself. In short, professional, specialized software is in short supply and the Color Computer user should be glad of any high quality additions to this market.

That is why Kage Engineering's release of a finite element structural analysis program is good news. Structural analysis means "Is a structure strong enough?" Can your roof withstand a snow load, or is the TV antenna mast adequate?

Finite element means to do the analysis by visualizing ("modeling") the structure as if it were composed of a large number of little springs, usually metallic. The method is in common use in the aerospace industry, and was used on NASA's space shuttle.

Of course, when simplified to a home computer, a number of subtleties are sacrificed. *Space Frame* does not include buckling or torsion effects, works entirely below the plastic deformation range, and defaults to steel unless told otherwise. (That would be minor for most users, but for us aluminum addicts it means overriding a default on each and every element input.)

As far as I know, this program has no competition, so it is *Space Frame* or hand drudgery for many engineers in private practice.

Kage Engineering claims it will run even on 16K. My CoCo is a 64K machine. I have used it not only for the test cases supplied, but even to analyze a fairly complex sphere I designed. The program can use the high speed poke, but I have only tried that on one simple model — it worked fine. Without the high speed poke, a 24-element, 16-node model took about 13 minutes to run. I would consider that time quite acceptable.

The program does retain three dimensions, which means it can do real world problems as well as teach engineering. The documentation is relatively complete, and the two examples will refresh the memories of engineers who are rusty on this discipline. I encountered two problems in using the program. On my most complex model, the node deformation table which printed out at the end was not consistent with the element deflections printed just above. That may indicate a bug, or may be a problem in my model of the structure, it is hard to tell.

My other problem was on I/O. *Space Frame* provides the capability of saving one's input to cassette at two distinct points in the analysis. The first worked as advertised; I never was able to recover data saved at the second point. I do not consider that problem serious because most users could discipline themselves to use the first option. Conversion to disk appears simple — I know how to, but not having disk I did not bother. The progam drove a Radio Shack DMP-200 properly with no problems.

Space Frame is supplied with a utility to enable 64K users to get 8K extra by disabling Extended BASIC. The utility worked properly, and would seem a good idea for most users because it is very difficult to tell in advance how much storage a complex model will require. For most users, the program will be adequate, but I would not suggest trying to model the Eiffel Tower on a CoCo!

So long as I have access to more elegant and larger versions free through my employer, I would personally have little incentive to buy this program. (I am licensed for private consulting practice, but am not doing any.) However, were I doing consulting — or even teaching at the university level — I know of no other program available in this field and I found *Space Frame* did its job properly with no major criticisms. I look forward eagerly to more specialized engineering software from a firm that clearly put together a good product.

(Kage Engineering, P.O. Box 3010, Lakewood, CA 90711, cassette $50)

— H. Larry Elman

Run CP/M With *Color Power II*

By Peter A. Stark

Originally written in 1973 for the 8080, CP/M is undoubtedly the grand-daddy of microcomputer operating systems. There isn't a 6809 version of CP/M, but it is still possible to run it on the Color Computer with the addition of the *Color Power II* module which, through a combination of hardware and software, puts a Z-80 microprocessor on the Color Computer bus.

CP/M has one feature which distinguishes it from the others — its tremendous popularity. This popularity has led to the availability of thousands of programs. While some of these may not run on the *Color Power II*, there is still a huge base of available CP/M programs, both large and small.

The Hardware

Color Power II (I'll refer to it as *CPII* from now on) is a metal box, about the size of Radio Shack's computer cassette recorder, which plugs into the expansion port on the right side of a 64K Color Computer. The disk controller, in turn, plugs into the right side of *CPII*.

CPII itself contains more than two dozen integrated circuits. To avoid loading down the CoCo power supply, *CPII* has its own supply in the form of an external power transformer that plugs into a wall outlet and connects to a jack on the back of the *CPII*.

Inside *CPII* is a Z-80 CPU, a 6845 video controller, 2K of video RAM memory, an EPROM character generator, and an assortment of unmarked TTL ICs. The entire assembly is very professionally constructed, with gold-plated connectors and a solid support for the entire assembly.

Since the *CPII* provides a display of 24 lines by 80 characters wide, in both upper- and lowercase, a good video monitor is needed. *CPII* does not provide an RF output like the CoCo's; a composite video output jack must be connected to the video input of the monitor.

The CP/M Disk Operating System

CPII comes with a copy of CP/M version 2.2. The *CPII* disk is supplied in a dual format — track one and part of track 17 are in Radio Shack format, while the rest of the disk has CP/M files. To start CP/M, you must type *LOADM "CP2"* and then *EXEC*. This loads a boot program, switches to 64K mode, and then loads and starts CP/M itself.

Once CP/M is started, operation will be very familiar to anyone who has used another DOS on the CoCo. CP/M has a number of built-in functions such as DIR (to display a disk directory), ERA (to erase a file), REN (rename a file), TYPE (display the contents of an ASCII file), SAVE (to save memory contents to the disk), or USER (which allows a sort of segmented directory structure). These can be run by just typing their names followed by any arguments that they need.

Typing any other command causes CP/M to look on the disk for a command file by the same name. If found, that file is executed. *CPII* comes with the following CP/M command files on the disk:
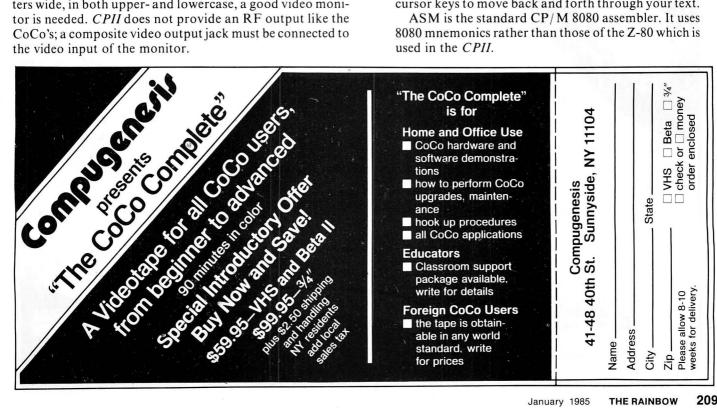
PIP is a 'Peripheral Interchange Program.' It is used primarily for copying one or more files from one disk to another, although it can also transfer files or data to and from disks, keyboard, display or printer. PIP cannot be used to copy files on a single drive.

STAT can be used to display the status of disks or files. For example, since DIR does not tell the size of a file or how much space remains on a disk, STAT can be used to provide this information.

SUBMIT and XSUB allow you to set up often-used combinations of commands as disk files and then call them with a single line command.

ED is the standard CP/M text editor. It is a line editor rather than a screen editor; this means that you do not see an entire screen full of text at one time, but only see the line you are currently typing on or perhaps the few lines directly above it, and you cannot use the cursor keys to move back and forth through your text.

ASM is the standard CP/M 8080 assembler. It uses 8080 mnemonics rather than those of the Z-80 which is used in the *CPII*.

LOAD is used along with ASM to convert the output of the assembler into a form which can be loaded and executed.

DUMP is a program for printing out the contents of a disk file in hexadecimal. The assembly language code for this file is also supplied, both for experimenting with the assembler, and also to show how user programs may be written to access CP/M.

DDT is the 'Dynamic Debugging Tool,' a debugging monitor program. It allows you to list or modify memory and start or trace programs.

CPII Implementation of CP/M

The *CPII* implementation of CP/M consists of two parts — the adaptation of CP/M itself and a collection of command programs for use with it. (This review describes version 1.5 of *CPII*'s CP/M software.)

Adapting CP/M to a new computer involves writing some code known as BIOS — the Basic Input Output System — which is the link between CP/M and the hardware of the computer. In *CPII* this code links the core of CP/M to the CoCo keyboard, display, printer port and disk interface.
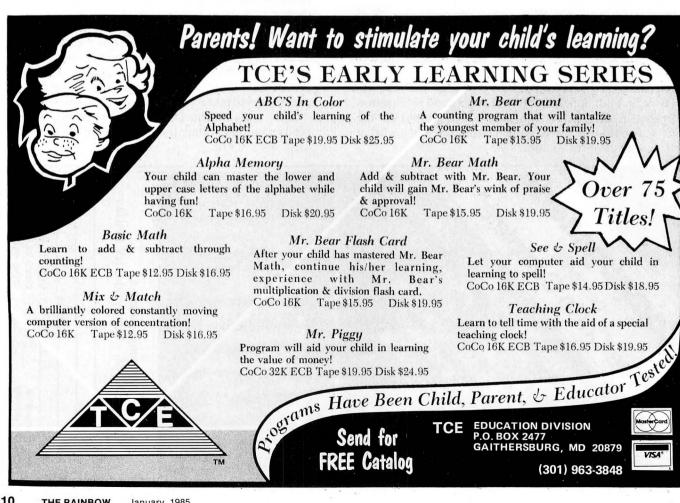
CPII lets the keyboard generate the full 96-character printable ASCII character set, plus control codes such as Control-C (used to restart CP/M) and ESC (used in many commercial programs). Many of these can be generated by single key closures, while some codes require that you press two keys at the same time.

As mentioned earlier, *CPII* includes a high resolution 24 x 80 display interface. Since this display is handled by hardware rather than software, it can provide excellent display quality if used with a good monitor. The video interface program which drives that hardware has been programmed to emulate the popular LSI ADM-3a terminal, so that programs written for the ADM-3a will drive the *CPII* screen in the same way. This is especially important for those programs which move the cursor around the screen. The only disadvantage in this case is that video display is much slower than we are accustomed to with regular CoCo BASIC.

CPII uses the standard CoCo printer port in the back of the computer. It allows operation at rates from 300 to 4800 Baud, with either seven or eight bits per character, and with either one or two stop bits. It also allows selection of whether line feeds will be fed through or not, and how the printer handshaking will be done. This covers a wide variety of printer configurations, both Radio Shack's and others'.

The disk interface portion of the BIOS is quite interesting in itself. According to the folks at Color Power Unlimited, they found that the Z-80, even running at 4 MHz, was not fast enough to handle the disk interface at double-density speeds, so they decided to let the 6809 in the CoCo handle that part of the job. The result, they say, is a complex blend of Z-80 and 6809 programming and hardware for which they have applied for a patent.

CPII uses the same basic disk format as CoCo disks — 35 tracks of eighteen 256-byte sectors each. This is quite different from CP/M's 128-byte sector, and so they pack two of CP/M's sectors into one CoCo sector. The result is a total of 146K of storage on one disk. (It is possible to use up to 40 tracks, but 80-track or double-sided operation is not possible.) A by-product of this disk format is that CoCo Disk

BASIC's regular *BACKUP* can be used to copy *CPII* disks, and *DSKINI* is used to format them.

In the area of disk-resident commands (CP/M calls them transient commands), *CPII* comes with four of its own command programs:

RESET quits CP/M and returns to Radio Shack Disk BASIC.

SYSGEN is used to generate a CP/M system disk for the *CPII* system. SYSGEN cannot, however, initialize a fresh disk, rather, it only modifies a standard CoCo disk for use with CP/M. Hence, to set up a new disk you must return to Disk BASIC (with RESET), use *DSKINI* to initialize the disk, and then return to CP/M and use SYSGEN to prepare it for use with CP/M.

CONFIG is used to change some of the CP/M parameters to fit the individual hardware used with the CoCo. It determines printer port parameters such as Baud rate (300 to 4800 Baud), number of data bits and stop bits, type of handshaking, and whether line feed characters are allowed or kept from the printer. CONFIG is also used for redirecting output from the printer to the screen or vice versa, and for setting disk drive parameters such as number of tracks (35, 36 or 40) and drive step rate (six through 80 milliseconds). Although CONFIG allows use of up to 40 tracks on a disk, the need to use BASIC's *DSKINI* command to format disks usually means that only 35 tracks can be used.

CONVERT is the largest and most ambitious *CPII* command. It permits *CPII* to read 18 other popular CP/M disk formats, including those written on Access, Cromemco, DEC, Eagle, Holmes, Kaypro 2, LNW, Lobo, Morrow, NEC, Osborne, TRS-80 Models III and 4, Xerox, and Zenith computers. Not all formats can be read since *CPII* cannot read single-density disks, 8-inch disks, hard-sector disks (such as North Star CP/M disks) or Apple II CP/M disks; nevertheless, the CONVERT program does allow access to a large variety of software from other machines. Note, however, that this is strictly a one-way street — CONVERT does not allow you to write disks which other machines can read. In addition, CONVERT does not let you read or write your regular CoCo Disk BASIC disks so there is no way to transfer data files, text files or BASIC programs to or from regular Disk BASIC.

For the user who wants some of the almost classic CP/M programs, Color Power sells copies of *WordStar, SpellStar, StarIndex* and *MailMerge*. Even when included with the *CPII* adapter, the total price is often less than the price of the individual programs on the open market. Due to a lack of documentation, I was not able to test these, but they do all seem to work. Also available is a communications program called *UltraTerm+*.

Documentation

CPII comes with a 29-page manual which covers the basic aspects of the *CPII* system and how to use it. This is a far cry from the two thick manuals which came with my copy of CP/M-68K (which runs on a 68000). The *CPII* manual leaves a lot of unanswered questions, and does not cover some programs (such as the ED editor, ASM assembler, or DDT debugger), at all. It suggests that the user purchase one of the many CP/M manuals, and gives a suggested bibliography.

User Comments

On the whole, the *Color Power II* system is a very solid piece of professional equipment. Aside from the lack of adequate documentation (which should probably improve with time), there is little one can seriously fault it with. Most problems noted were small ones, typically the kind that show up in any new product.

For example, *CPII* allows a fairly good implementation of *WordStar* and *SpellStar*. But a small problem occasionally causes this combination to scroll an extra line, so that when *SpellStar* comes back to rewrite the screen it writes one line lower than it should, thereby causing some confusion. Also noted was the fact that the BREAK key, which usually can break any program, does not work within *WordStar*.

The lack of a BASIC interpreter is also a small problem, especially since CP/M cannot read or write CoCo disks. (Since every other DOS for the CoCo has such file convert utilities, I am sure it's only a matter of time until they are available for *CPII* CP/M as well.) As a test of the CONVERT program, I got a public domain CP/M disk prepared for the Kaypro 2 computer. Though I was able to convert and read all of it, it was a challenge to figure out a way of using the BASIC programs on it. (I eventually transferred them to another computer via the serial printer port.)

(Color Power Unlimited, Inc., P.O. Box 606, New Providence, NJ 07974, $329. With *WordStar* and *MailMerge*, $498; *SpellStar* and *StarIndex* $79; *UltraTerm+*, $55.)

RAMDOS Is A Good Disk-Type Tape Program

RAMDOS is a tape-based utility program designed to use part of available memory as a storage area to stack several programs while the rest of memory is active. The machine language routine allows for the user to departmentalize the memory by use of a *CLEAR* statement and an *OFFSET* to protect a portion of RAM. The program provides for a disk-type operating system to facilitate the movement of programs from or to storage or active RAM. The intent is to be as close to using this storage RAM as a virtual disk. The method used by the program author is that the user moves into the program routine by entering an *EXEC* and then selecting the appropriate command for execution and then leaving the routine to return to the active RAM area again.

The disk-like operating system of the program includes commands to save a program from active RAM to storage RAM; list all the program in storage; rename a program; kill a program; clear all programs from storage; a 'help' menu which list the commands; a 'Q' command which returns the user to BASIC; and a 'restore' command which transports a program from storage to active RAM. All of these commands acted on the programs without trouble and faster than the blinking of the cursor!

The number of BASIC programs that might be stored depends on the length of the individual programs (and the amount of space reserved for storage). In a typical configuration of several short programs, each may be loaded sequentially in the storage RAM. The user then has the option of calling up any of the stored programs to the active RAM. In the transfer from storage to active RAM, the active RAM is wiped clean, thus this system was never intended to pass data between programs, or act as command stream system, or provide for linking or chaining of programs.

As noted earlier, the system works best with a group of short BASIC programs. Machine language programs may be used in this system, but such machine language programs must be assigned specific locations through the use of "offsets" at loading time so as not to overwrite the host program. The routine has trouble handling very long BASIC programs because of the memory limitations imposed by dividing the RAM into an active and storage segment. The division of RAM space is accomplished by use of *CLEAR* statement and loading *RAMDOS* with an *OFFSET*. The selection of a coupled set of *CLEAR* and *OFFSET* statements which provides for allocation of the RAM is the problem. If the user selects a combination that allocates most of RAM to storage, then the user might encounter the OM Error when trying to *CLOAD* a long program. Likewise, allocating too little RAM to storage will result in an OM Error if this large program does not have enough room to be stored.

The program, once loaded, functions smoothly and is fast. The selection of the proper sequence for loading with *CLEAR* and *OFFSETs* is the hardest part of the whole operation. The manual offers an example in which the user is instructed to *CLOADM RAMDOS* with an *OFFSET* then enter *CLEAR,200* with a number one less than the *OFFSET*. With that example the user is left to his own experimentation. One other instruction is offered in which the manual indicates that if the user has Extended BASIC, then a *PCLEAR 1* is necessary before loading the program. (Interestingly, one of the long programs used for the test required a *PCLEAR 4* and after following the manual's suggestions, the program when returned to active RAM would not work.)

The manual's major shortfall is not being more informative on the loading patterns to be tried for different needs or RAM size. The manual's title page notes the program will run on 16, 32 or 64K computers. While true, it should be noted that it is compatible with only the lower 32K of the 64K equipped computers. The manual suggests that some bank switching programs might work with *RAMDOS* if the jump table is maintained. No test of a bank-switching routine was attempted.

RAMDOS might be of interest to non-disk users who might want to use part of the RAM as a "virtual disk" storage area. Within the limitation of the size of the individual programs and designated RAM, the program might fill the need of a disk-like addition.

(Dataman International, 420 Ferguson Ave. North, Hamilton, Ontario, Canada L8L 4Y9; cassette $25.95 U.S., plus 3 percent S/H, $2.50 minimum)

— Robert Foiles

Super Disk Utility —
A Collection Of Useful Tools

"Where is the beef.dat?" If you had used the Catalog option of *Super Disk Utility*, all that would be necessary would be to look down the printed list to learn where the "beef.dat" program was located.

Cataloging all of the files on all your disk is one of the nine operations grouped on the *Super Disk Utility* program sold by Petrocci Freelance Associates on an unprotected disk. The other operations supported are *Diskzap; Date Disk Files; Purge Files; Copy by Files; Directory Sort; Super Directory; Diskzap 40;* and *Track Formatter.*

The program author has arranged access to seven options through a main menu system. When a selection is entered it is loaded from the disk and automatically runs. The options have submenus to offer further functions. The other two programs on the disk *(Track Formatter* and *Diskzap 40)* are machine language programs that are loaded independently as needed.

From the main menu the user can select the drive number (0 to 3) for default drive. This option is very useful for the two drive user since it allows the program to move through the options without several disk swaps. The single drive user is given prompts as to when to switch disks (system disk or data disk) and switching disks is necessary each time a major option is selected.

Operating the Catalog option is simple for either single drive or two drives. The program prompts the user to enter a name for the disk which will have its directory read into memory. The disk is read and the next name is entered for the next disk, etc. When the user has run out of disks to read (or reaches the program's limit of 800 files) the hard part of the job is done. The sorting of all the disk directories into order by either "file name," "disk name" or by "extensions" is fast. The sorted file is automatically saved (with the program assigning the extension for each of the three types of sorts). For this review a total of 30 disks were read into the catalog. These disks produced 330 files which were sorted in less than one second. The sort routine is so fast that I had to run the test twice to be sure the sort actually took place. The only way to use the results of all this fast action is to have the listing of the sorted files printed out. Again, the selection of either "file name," "disk name," or "extension" is necessary for the printout of the records by that option. Once a catalog has been constructed, the program allows for modification by either deleting files or adding to the "inventory."

Diskzap, the other "major" program in the package has its own submenu. The options provide for "allocation table check," "verify disk sectors," "display file sectors," "display disk sectors," "move" or "copy disk sectors," "covert granule #" into track and sector or "zero disk sectors." During any of the display options, the program provides for modification of any location shown on screen by moving the cursor to the spot and typing over the desired notation. Once changes are made to suit the user, hitting ENTER will save that change to the disk. This section of the program works well and a change can be entered faster than some other "zap" programs.

The *Track Formatter* program is able to reformat any selected tracks (up to 40 tracks) on a disk with problems or it can format an entire disk. By using this program a not yet formatted disk can be given the same status as a disk given the *DSKINI* pass; but the program then will go back and verify all the tracks; and in a fraction of the time required to use the *DSKINI* function alone. Another feature of this program that may be of more interest to users of non-Radio Shack drives is that the "Sector Skip Factor" is controllable.

Diskzap 40, according to the program author, allows accesses to 40 tracks for those users not using Radio Shack equipment. This program was not tested since the system used for this review was standard Radio Shack equipment (which only has 37 tracks). The program also allows for a shifting of a copy of a disk's directory to track 37 as a "cash protection" option. Naturally, the directory restoration option is also included.

The 13-page manual covers the programs and indicates those areas where problems might occur. It is clear and easy to follow. Page 14, however, is printed in such a small print that it is almost unreadable; but it appears that page is the "End User Agreement."

The *Super Disk Utility* is a collection of some interesting and useful tools and some functions that may not be of interest to some users. The ability to purge files, copy or date a disk file, as well as the Directory Sort option which will read the directory, sort in alphabetical order or group by extensions and rewrite to disk, all from within the program, are there if you *want* them.

However, the Catalog option has a fast sort function; but there is no search or find function included. Thus, the speed of the sorting is outweighed by forcing the user to go to the slowest operation of the system — the printer — to make use of the information. There are other Catalog programs on the market which include a search function and sell for a fraction of the cost of this program.

This set of disk programs may appeal to some users as it is offered.

(Petrocci Freelance Associates, 651 N. Houghton Road, Tucson, AZ 85748, disk $29.95)

— Robert Foiles

One-Liner Contest Winner . . .

Guaranteed to drive you up the wall after a minute or so, this program draws a circle that gets larger and smaller over and over!

Michael Rosenberg
Prestonburg, KY

The listing:

```
Ø PCLEAR8:FORX=1TO8:PMODEØ,X:PCL
S:CIRCLE(128,96),X*1Ø:NEXT:FORX=
1TO999999:FORY=1TO8:SOUNDY*2Ø,1:
PMODEØ,Y:SCREEN1,1:NEXTY:FORY=8T
O1STEP-1:SOUNDY*2Ø,1:PMODEØ,Y:SC
REEN1,1:NEXTY,X
```

(For this winning one-liner contest entry, the author has been sent copies of both *The Rainbow Book Of Adventure* and its companion *Rainbow Adventure Tape.*)

Star Trap Game Promotes Cooperation Among Children

Star Trap is a new and different game for one or two players aged seven or older. It is produced for Radio Shack by the Children's Computer Workshop — an activity of the Children's Television Workshop (the producers of *Sesame Street, the Electric Company,* and *3-2-1-Contact).* Supplied on cassette, this game requires Extended BASIC, and joysticks.

Before describing the game, I'd like to tell you how it differs from most games. First, this game teaches cooperation, rather than competition, when played by two children. The two players work together to accomplish the game's objective — to trap the computer-controlled star.

A second difference is the cassette format. Side one contains the "standard" game, ready to be loaded and played. Side two contains a modified version which allows the player(s) to "customize" the game via an easy-to-understand menu. Selections include the game (maze) size, background color, player and star speeds, active gates (explained below), and several miscellaneous options: the number of X's (used to trap the star), whether blockers are desired, the number of players, and the presence or absence of the maze walls.

The object of the game is to trap a single star — to block its movement either with X's (placed at the player's position

when the joystick button is pressed) or by cornering it against a wall or between two players. The score depends on the time required to trap the star and the number of X's used — the shorter the time and the fewer the X's used, the higher the score. There are six gates which affect the players and star differently when passed through:

1) The X gate replenishes a player's limited supply of X's.
2) The speed gate increases the speed of the object passing through it — whether a player or the star.
3) The slow gate decreases the speed of the object passing through it.
4) The jump gate causes the object entering it to exit at a different location.
5) The invisible gate renders the star invisible for a short time after it passes through.
6) The magic gate allows the star to pass through walls for a short time.

When playing the customized version (on side two of the cassette), each of the gates can be included or excluded via the menu.

To further complicate play, blockers appear periodically at random locations for a brief period to block both star and player movement. (These blockers also can be included/excluded via the menu when playing the customized version.)

A detailed instruction book includes step-by-step procedures to assemble the necessary hardware, to load the program from cassette, and to play the game (using either the standard or customized versions). Possible difficulties and typical solutions are included throughout the book. All game symbols are explained and player strategies are outlined. The final pages suggest several related games that don't require a computer.

As you have probably inferred, I'm impressed with *Star Trap* — its unusual format, its cooperative nature, and its complete instructions. It encourages children to modify the game, possibly piquing an early interest in programming. Radio Shack has a winner in this game.

(Radio Shack stores nationwide, on cassette for $19.95)

— **Jerry Oefelein**

One-Liner Contest Winner . . .

This is a very "graphic" demonstration of how things drawn in one *PMODE* look when switched to another.

Bill Valentine
Fort Worth, TX

The listing:

```
Ø PMODE RND(5)-1,1:SCREEN1,1:X=R
ND(128):Y=RND(8):HW=RND(185)/99:
CIRCLE(126,96),X,Y,HW:PAINT(128,
96),Y,Y:GOTO Ø
```

(For this winning one-liner contest entry, the author has been sent copies of both *The Rainbow Book Of Adventure* and its companion *Rainbow Adventure Tape.)*

Learning To Count
With *Counting Things*

Two programs helping children learn to count and recognize numerals are presented on the tape, *Counting Things*. Instructions and guidelines for the parent or teacher are included as a separate, introductory program, which advises the adult to explain the programs first, demonstrate their operation, and pose questions which can be answered by using the programs.

The first program, *Counting Things*, asks how far the child wishes to count and then goes to a Hi-Res screen showing three vertical lines labeled 100, 10 and 1 and, in the lower half, zero is spelled out and a 0 is shown on the screen. Under this, the computer asks, "What is the next number?" When the numeral one is pressed, the bottom portion of the screen shows the number one and spells out one, and, in the upper portion a plus appears and one small horizontal line is drawn on the vertical line in the units column. The program continues in this manner, adding lines to the units place, until the number 10 is reached, when a small line is placed in the tens column and the lines removed from the units

column. This visualization of the number line, the place holding property of zero, and demonstration of how the number system works in moving from the units to tens to hundreds places is valuable, particularly if an adult has posed one of the suggested questions so that the child's attention is drawn to this feature of the program.

The second program is similar, although this time the child first enters how many numbers he knows and the computer then places numbers of pluses on the screen and draws corresponding numbers of small lines on the vertical lines, requiring the child to enter the numeral which the pluses represent. Both programs respond only to the correct number.

Thompson House wisely points out that these programs are not intended for repeated use to gain mastery, but rather to reinforce learning at various stages. At first the child learns to press the next number in the sequence, and becomes familiar with the written word for each number, then he counts objects and gains an understanding of the number line and perhaps some better understanding of the base 10 number system.

(Thompson House, P.O. Box 58, Kamloops, British Columbia V2C 5K3, 16K ECB tape $14.95)

— Carol Kueppers

Software Review

Pak-Panic — The Old Game With A New Twist

With centipedes, monsters, invisible mazes and ghosts that can go through walls, *Pak-Panic* from Tom Mix Software is unique compared to all of the competition. *Pak-Panic* is a 32K 100 percent machine language, arcade-style game that uses the left joystick and firebutton.

The scenario is as follows: You are Pakman. Your job is to go around the screen eating dots, power pills, and bonus prizes while avoiding monsters.

A power pill is one of the larger dots on the screen. Seven are on levels one through four. Six are on levels five through nine. When a power pill is eaten, Pakman has the power to eat all of the monsters he pleases. Whenever Pakman eats a monster, his ghost appears at the top of the screen. When seven ghosts have appeared at the top of the screen one of two things will happen. Either one of the ghosts will come out and float around the screen (even through the walls) hunting for Pakman, or the seven ghosts will link together to form a centipede that will do the same thing. Even with power pills, Pakman cannot overpower ghosts or centipedes.

Bonus prizes appear in the middle of the screen about twice a board. When bonus prizes are eaten they are stored in a box below the screen. When 14 prizes have been eaten you get a bonus of 14,000 points. This can only happen twice.

Four more tricks the programmer threw in to make the program better are invisible mazes every four rounds, the ability to store power pills, a selection of difficulty at the beginning of the game, and a high scores board.

You can store power pills by eating a pill while a previous pill is in effect. Stored power pills can be used by pushing your button while no power pill is in effect. A maximum of six power pills can be stored at any time.

I liked *Pak-Panic* and I think many other people will like it.

(Tom Mix Software, 4285 Bradford NE, Grand Rapids, MI 49506, tape $24.95, disk $27.95)

— Pat Downard

Test Manager — A Useful Addition To Your School/Home Library

Test Manager is a program that allows you to create files of multiple-choice questions, then select specific questions to be printed on a test. An answer key is also printed with each test you prepare. *Test Manager* requires a 32K disk system with a printer. The program will work with one-disk drive with a minimum of fuss, but a two-disk system will make life easier if your files contain more than 75 questions. Question files can contain a maximum of 150 questions.

This program works exactly as described in the documentation. The documentation consists of six pages of information, clearly worded and neatly printed. The program worked without flaws for two weeks of rather intensive use. I made up a variety of tests, using all the options on the menu. One of the especially nice features of a program of this type is that it allows the teacher to create one set of questions (say 75) and then create five or six different versions of the test, each with its own key. This, of course, cuts way down on the chances for cheating, as well as allowing for individual differences in students or classes.

You begin by setting up your file of questions. Each question can be up to 245 characters long (I found this to be more than adequate.) A bracket is used on-screen to mark the end of the space allowed for a question, a very handy feature. If you do exceed the limit, the program tells you so, erases your question, and invites you to try again. Previously entered questions are not lost. After each question is entered, you enter the answers, correct answer first (the answers will be presented in true random order when each test is printed).

You can create true-false tests by just entering two answers and entering a blank for answers three and four. Answers can be up to 180 characters long. This allows for complete thoughts to be used rather than just one- or two-word answers. You may go back and add, delete or change questions later if you wish. The only restriction to this is that you cannot delete a question from a file containing more than 75 questions unless you are using a two-disk system. Both single and two-disk systems will hold a maximum of 150 questions in a file. You may also merge two different files together to make a new file.

Next, you select the questions you wish to use on a test. These can be from more than one file. The process of selection creates a new file of your selected questions, so you need to be careful not to try putting this new file on a nearly full disk — another place where two-disk systems make life easier! You then print out the test. You are asked for a name for the test, the form code (so you can match the correct key to the test), and the name of the file that will be used in printing. The program will also print a space for the student's name and class. You have the option of printing the questions in the order you selected them, or having the computer print them in random

order. The answers are always printed in random order, even though you entered the correct answer first. The page break will always be between questions.
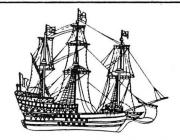
The printed copy of the test is neatly arranged, ready for photocopying. Or, since nearly all printers using friction-feed can print directly on a ditto master if the ribbon is removed before printing begins, you will have a letter-perfect test and no purple ink on your fingers! (See Valerie Rhead's article on Page 140 of the September issue of THE RAINBOW for a full discussion of this.) You have the option of single sheet printing (useful with the dittos) or continuous printing.

The only fault I found with *Test Manager* is that it isn't very economical with paper. Even with one line questions and one word answers, the most questions you can get on the first page is seven (eight on subsequent pages.) For long tests, you will have a number of pages to ditto or photocopy. If your printer supports a condensed-print mode, this may help. You could also ditto pages back to back, and some photocopy machines have this feature as well.

Overall, *Test Manager* is a well-developed program that does just what it claims to do. It would be a useful addition to your personal or school's software library.

(80 Custom Software, 5720 Brooke Lane, Sylvania, OH 34560, disk $29.95)

— Mark Williams

Key-Wiz — Easy, Fast, Useful Database Manager

Key-Wiz for 0S-9 is a simple, useful utility, especially for those who don't have the money for a $200 database software package. This program "is a multikey search utility designed to quickly search a file of index cards and find all entries which match a profile of keywords" and "logical operators such as AND, OR, and NOT to provide a very selective search criteria." This simply means it will find "entries" in a simple text file by searching for "phrases" or words you specify. An entry is nothing more than one to 60 lines of text 80 characters or less. Each entry is separated by a blank line and nothing more, so any 0S-9 utility that can create such a file, such as a word processor, can create your database.

Supplied with *Key-Wiz* is a sample database containing two lines per entry. The first line contains a book title and author, the second line contains a publisher's name. When invoked, *Key-Wiz* will present you with a menu of options. You can specify input and output files when you execute *Key-Wiz* or you can select the 'I' and 'O' menu options to specify those files after executing *Key-Wiz*. These assignments can be changed at any time while in the utility. Your output file is, by default, your screen and can be changed to a disk file, or perhaps to your printer. You can also select options to limit the search to any number of lines. For example, if you know the word(s) you are searching for are contained only in the first four lines, and each entry is 15 lines, you can tell *Key-Wiz* to search *only* the first four lines of each entry, thereby speeding up the search considerably. And *Key-Wiz is* quite fast, considering the work it must do. You can also limit the number of lines output for each match, thereby formatting your output to show only the data you wish to see. You may also tell *Key-Wiz* to display the output continuously or to pause after each entry displayed, waiting for you to press the ENTER key for the next entry. *Key-Wiz* also comes with *Sort-Wiz*, a quick sort utility designed to sort the entries on any line you choose. This is desirable for sorted output, and the standard OS-9 sort utility, if used, would sort the blank lines to the beginning of the file. This would destroy the entry markers in your file (the blank lines), hence *Sort-Wiz*.

I found one annoyance and one bug in this otherwise excellent utility. The annoyance is that there is a useless "Help" option, which does nothing more than redisplay the menu. Simply pressing ENTER does that already. It may make the program too large, but it would have been nice to have a brief "help" for each menu choice. The bug I found occurs like this: I've opened my input file and entered my search criteria. Entries which match are being displayed just fine. After seeing about 10 entries, I don't wish to see the remaining possible hundreds of matches. How do I stop the search? If I use BREAK, I interrupt the program completely and am dropped back to 0S-9. My CHD and CHX directories revert to bootup default! If I use CLEAR BREAK, the display continues, and when done, I see the menu constantly being redisplayed and it doesn't stop! The only thing I can do here is press BREAK and completely terminate the program! There is definitely a bug. There is an 'E'xit command in the menu which works perfectly, and even closes the files, but that is for "normal" completion.

Overall, for the price it is an excellent little database utility, again, for those who do not need or cannot afford the more expensive, though more powerful, database management software.

(Interactive Micro Systems, P.O. Box 21007, Columbus, OH 43221, disk $24.95)

— **William Van Nest**

Howard Medical Disk System Offers More Storage

Many CoCo users are no doubt aware that several of the more popular personal computers (such as the IBM PC and PC*jr*, and Radio Shack's Model 1000, 1200 HD and 2000) use double-sided disk drives to cram twice as much data on each diskette by using the "other" side of the disk. All diskettes that I know of have a magnetic coating on both sides (unlike recording tape, which is almost always single-sided), but so-called "single-sided" disks are only guaranteed to work on one side. In practice, single-sided blank diskettes generally work fine with double-sided drives. Howard Medical's *Drive 0 Package* lets you get up to 360K on a single diskette on the CoCo, and is also able to handle standard CoCo disks.

The *Drive 0 Package* consists of a DD-2 double-sided disk drive (with case and power supply) and a J&M Systems disk controller. A special cable is also included. The disk drive is a TEAC half-height unit, the same one used in the Sanyo MBC-550-2/555-2. Its case is sturdy, though plain, and the power supply is well able to drive it.

The JDOS operating system supplied with the controller is the key to using double-sided drives on the system. It is capable of using either 35 or 40 tracks and either one or both sides. The J&M controller is also configured for double-sided drives. All you have to do is format your disks on both sides in the J&M format, like this:

DSKINI 0, 2, 40

Disks formatted in this way will have 360K of space available. JDOS automatically adjusts for the format of whatever disk you use, including standard Radio Shack disks.

One problem with using JDOS is that a few programs (such as *Telewriter-64*) are very dependent on Radio Shack Disk BASIC features and won't run with the entirely different JDOS ROM. If you have a Radio Shack disk controller, you can use it with the new drive with no problem (but only on Radio Shack-format disks). If you have 64K, another way would be to use the SYS SAVE and SYSTEM programs in Part I of "Cooking With CoCo" (August 1984, Page 18) to save Radio Shack Disk BASIC on a special disk and use it in place of JDOS if needed.

If you think that 360K per disk will help you use the CoCo more effectively, the *Drive 0 Package* is a good buy.

(Howard Medical Computers, Box 2, Chicago, IL 60690, $395 plus $7 S/H)

— Ray Edwards

Calorie Counter — Weight Analyzer Is Just What The Doctor Ordered

Draco Software's *Calorie Counter — Weight Analyzer* program is just what the doctor ordered. This tape-based program offers a well-rounded choice of over 400 food items to use in daily menu planning, with guidelines given on how best to plan your diet program.

A seven-page guide and instruction booklet outlines the basics to allow the user to lay out a regular, well-balanced diet for children, teens, adult males, adult females and lactating females. All that is missing is a guide for pregnant females, and perhaps it is intended that, in such cases, direct medical supervision of diet is called for.

Once the program is *CLOADed*, *RUNning* should get you going. (I have an older CoCo with the *PCLEAR* problem, and so had to type *RUN* twice.) A brief quiz allows the computer to determine your correct weight, and displays it on the screen. You should make note of this, as you will need it later when compiling your daily food selection. Something that might be done in a future edition of the program is to check if a machine is 64K and if so, *POKE* the maintenance calorie level into high RAM. Then when the next section is loaded, it could check and

PEEK the information into the program, to be displayed for reference. A minor point is that age is not considered in determining the calorie level needed to maintain body weight. While the difference may not be great, a 20-year-old of a given size and weight will burn more calories than an equivalent 50-year-old.

A very few items needed further clarification of serving size (chef salad, raw carrot) but in almost all other cases the serving is given in standard cup or ounce increments.

The ability to store a daily menu on tape for future use allows the user to build a "file" of menus which, with a little planning, would allow a dieter to plan a varied meal plan and cut down on those moments of indecision which might otherwise lead to the kind of snacking that can lay the rest of a day's good intentions to waste.

A nice option, should you succumb to temptation, or if you have a food not on the listing provided by *Calorie Counter*, is one which allows you to enter an extra calorie amount. This allows you to account for (my favorite) ice tea with extra sugar, or other personal foibles.

A printer option gives the user a hard copy to keep by your refrigerator or to take to the store. This can help reduce nibbling, as well as cut down on impulse buying.

Both fun and (sometimes) depressing to use, Draco's *Calorie Counter* should prove a good buy for any dieter.

(Draco Software, 22 Cassell Street, Portland, ME 04102, $29.95)

— Nevin J. Templin

Talking Adventure Starter — A Vocal Experience For Adventurers

By Kenneth D. Peters

Welcome to Adventuring with voice responses! *Talking Adventure Starter Package* by Owls Nest Software consists of two Adventures, *MYHOUSE* and *PIRATES,* and a two-page documentation designed to help teach Adventuring to the beginning Adventurer while providing the unique experience of vocal responses throughout the Adventures. As an added attraction, these Adventures are relatively simple, nonviolent, and fun; suitable for very young children of reading age.

Owls Nest has two versions of the *Adventure Starter Package.* The standard version was reviewed in the February '84 RAINBOW and has only a text response. The *Talking Adventure Starter* is meant to be used with Colorware's *Real Talker* voice pak, introducing you to Adventures with the added realism of voice responses.

The reviewer of the standard *Adventure Starter* mentioned having to remove the disk controller and enter *PCLEAR 0.* If that was a problem then, apparently both conditions have been improved. I found both Adventures in the talking starter package loaded with a simple *CLOADM* operating under Disk BASIC. The actual Adventures are written in BASIC, but use *CLOADM* to load and auto execute and also to provide an initial screen to read while the Adventure loads.

I was disappointed by the lack of an introductory voice when the Adventures first begin. In *MYHOUSE,* you are given an Adventure welcome (in text form), with the object of the Adventure being to find the hidden golden coin and return to the front porch. (First you must find a way to enter the house.) I anxiously waited for a voice to come forth and welcome me while I was trying to decide what to do. None came. "Where's the voice? I thought this was a talking Adventure." My initial disappointment passed as I progressed through the Adventure. With increasing interest and enthusiasm I discovered that the voice pak was used primarily for responses to the Adventurer's commands: East, West, North, South, Help, Get Paper, etc. Actually, my kids and I found that the voice talked to us quite often as we explored the house using one- or two-word commands, such as "Open Door" and "Read Paper". Moving through the house by entering compass directions (N,S,E,W) and Up and Down sometimes provoked a vocal response. Sometimes the Adventure voice provided us with a humorous response to our commands and movements. Sometimes it gave us some useful clues. Overall, use of the voice pak gave us a remarkable experience not to be soon forgotten.

One thing about the Adventure that bothered me was the slow character print on the text screen. *MYHOUSE* has a poke in it that reduces the character printing rate across the screen. I found it annoying to have to wait so long to see what the response and changes were on the updated screen. However, I am not a beginner in Adventures and after thinking about it, I realized that

feature could be a big plus in helping draw the attention of beginners to screen updates and the fact that changes may have taken place as a result of something they just did, but may not be obvious unless they actively look and review their surroundings. In *PIRATES,* the poke has been modified so the screen is updated much faster. (Being written in BASIC, both Adventures can be altered to delete the poke if desired.)

Lack of a vocal welcome and the slow screen update are minor. I think the *Talking Adventure Starter Package* does a super job of developing an interest in Adventure game playing and in providing the beginning Adventurer with many detailed hints, clues, and ideas about playing Adventure games. I watched my two older children, ages seven and 10, play *MYHOUSE* with the same engrossing and endless enthusiasm as they play the arcade quality games on the CoCo.

In addition to the excellent documentation containing all the helpful hints and ideas, both Adventures use the HELP command. HELP is used extensively in the first Adventure, *MYHOUSE.* Typing HELP invokes a vocal response ranging from giving you moral support to suggesting new ideas to try or things to look for, to telling you simply that nothing will help you here. *MYHOUSE* is relatively simple, meaning there are few rooms, movements, and obstacles to overcome and contains no death traps. Additionally, *MYHOUSE* is similar to your house and mine — a setting we all have some familiarity with. In the eyes of a child or beginning Adventurer no Adventure may seem simple at first. I found *MYHOUSE* quite enjoyable and it introduces the basics of Adventuring very nicely.
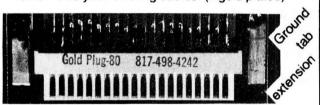
PIRATES is also relatively simple, when compared to other complex Adventures that take days and weeks to complete. There are relatively few moves (rooms/scenes) in *PIRATES* and they are easily mapped out. But *PIRATES* is much more involved and complex than *MYHOUSE,* and contains several death traps. Your goal in *PIRATES* is to recover the pirate's treasure. *PIRATES* introduces a greater depth and exposure to the subtle ideas, increased number and variety of verbs, the importance of doing the right thing at the right time, and the persistence needed at times to obtain a desired result and finish an Adventure.

Both Adventures have a game save feature. I used the option in both Adventures to make sure it worked; however, some Adventurers may find they do not need it. I finished *MYHOUSE* Adventure within an hour. My children both finished it within a half-dozen sittings (one-half hour to one hour per sitting). *PIRATES* took considerably longer but is possible to complete in one evening unless you really get stuck.

I thoroughly enjoyed both Adventures. I keep reading or hearing about many people who get discouraged and frustrated by Adventures and give up or lose interest. I believe both programs and the accompanying documentation provide enough help and foster sufficient interest to ward off such feelings. At the very least, *Talking Adventure Starter* would probably give the unsure beginning Adventure player a truer feeling for what Adventuring is about and an opportunity to enjoy and learn more about Adventuring.

If you've been contemplating Adventures or if you've tried Adventures and have become frustrated with them, I would recommend giving *Talking Adventure Starter* a chance. I also would highly recommend the *Adventure Starter* as an ideal nonviolent Adventure for young children, age six and up (reading age). The children love it and you might even consider a joint parent-child partnership in solving the Adventures. Even if you are beyond the beginner's level, I think you'll find these two Adventures enjoyable. Both are possible to finish in an evening and both will give you a unique vocal experience. By the way, if you don't have Colorware's *Real Talker,* the standard text *Adventure Starter* is also available.

(Owls Nest Software, P.O. Box 579, Ooltewah, TN 37363, $17.95 tape)

Play With Language Is Exhilarating, Entertaining And Educational

By Michael F. Garozzo

It is hard to believe how far computers have developed in the world of education. After all, it's not *that* long ago I attended school. Actually, of course, education is an ongoing experience, and we all are continually learning how much we do not know. But I certainly do not remember school being as much fun without computer programs like CCW's *Play With Language*.

Children's Computer Workshop, in case you do not know, is part of Children's Television Workshop, the group who produced *Sesame Street* and developed Sesame Place, the educational amusement park in Pennsylvania. As a matter of fact, the computer programs that CTW developed were originated at Sesame Place. Living about 10 minutes from Sesame Place, I can verify the statement in the introduction of the booklet that says, "Fifty-seven computer games are available . . .," my children and I have played most of them.

I think the three programs included in *Play With Language*, developed from the low-resolution, fairly simple games at the Park, are amazing. The 35 people listed under the titles of CCW's print development team and software development team deserve the credit for a job well done and I feel that it was honorable of CCW to list their names.

Play With Language is designed for elementary school use. This is not to say a family would not profit from the learning experiences; however, the format of the instruction booklet is definitely classroom oriented. The information begins by telling you that the program includes three language-arts learning modules. Each learning module contains the diskette, 10 reproducible worksheets, five activity cards, one gameboard and one poster describing how to play the activity. An extensive teaching guide states ". . . and ancillary materials to provide for individualization of instruction, child/computer/teacher interaction and an educational philosophy which encourages active participation in reading and writing." And this was in the first paragraph. Does this scare you? Would Cookie Monster refuse a chocolate chip cookie? The general information section begins with the word "Relax." The booklet is designed for the beginning computer user as well as the more experienced. The instructions are precise (and so easy to follow that Oscar the Grouch could not get mad). And in case you just unpacked your Color Computer for the first time, "Setting Up Your System" describes everything you need to know right down to which wire to plug into what connection.

The three, separate programs with the *Play With Language* series are: *Picture Place!, Roll-A-Word,* and my. personal favorite, *Bagasaurus.* According to the instruction booklet all programs have the same affective and cognitive goals.

The affective goals are:

A) Students talk and exchange ideas with their peers and teacher as they interact with the computer.

B) Students are involved in self-directed learning as they explore environments and think about information in new and different ways.

C) Students have fun and enjoy their experiences with the computer, as they play the activity.

The cognitive goals are:

A) Students practice sight words and beginning vocabulary as they exchange words and their picture referents.

B) Students practice making words in rhyme patterns and complete sentences in clever ways.

C) Students practice several reading comprehension skills and create stories.

In addition, each program has an individualizing menu for the teacher called "Learning Manager"; a "Teaching Extensions" section in the booklet for additional classroom activities; a stand-alone "Gameboard" to use in the classroom with a group; and a "Dictionary" that lists all words in the program.

Picture Place! has 193 word-pictures and six background scenes. Each time the game is played, " . . . the computer randomly selects 24 words from a pool of 40 words associated with the chosen background scene." The student picks a word from a list of four words shown, hits the ENTER key, and using the joystick, moves the word-picture onto the scene. When the student feels that the proper location has been reached, the ENTER key is hit again and the word-picture is placed upon that spot. The bottom of the screen has a list of four words, the word "more" and the word "end." The joystick glides a rectangle across

the screen for placement of the words on the scene or encircles individual words on the list to be chosen. When you encircle "more" you get the next list of words, if you encircle "end" you stop the program. It should be noted that the description I am giving you does not give justice to the excellent high-resolution stage scenes or to the superb machine language music used in the program. *Picture Place!* is designed so the teacher can develop special skills with short vowels, long vowels, blends, the silent 'e' rule and syllables. As with all the programs, the teachers can create their own lessons by choosing certain picture-words and saving them to a disk. Of course, the teacher can save each student's creation on a separate disk. As before, the instruction booklet is very precise in explaining each and every step. The "Teaching Extensions" section includes 10 activities that correlate with the program. In addition, there are five activity cards for the students. Each activity card is a 5" x 7" hard, glossy card that is designed to withstand classroom use. The "Gameboard" is a Monopoly-style game that is printed on glossy paper. It would probably be worthwhile to glue it down to a piece of cardboard for permanent use. Complete instructions are included in the booklet. The 10 reproducible handouts include a cartoon (for coloring), questions and an activity.

Roll-A-Word is a fascinating way to have students match beginning and ending sounds. The more complex version has the students using blends. A sentence is shown with blank spaces. By using the right and left arrows, pictures are moved across the top of the screen. When the student reaches a picture that matches the sentence, he uses the up and down arrows to roll the letters in the rectangle.

Upon matching the letters to the proper picture the student hits the ENTER key. If correct, the picture either rides, flies or floats to the correct position on the sentence. If incorrect, he has to continue. It is as much fun rolling the letters and pictures as it is getting the right answer. As before, the program has a learning manager so the teacher can create other versions of the program. Student sentences can be saved to disk if desired.

There are also 10 "Teaching Extensions," five activity cards, and a gameboard. The *Roll-A-Word* dictionary is divided into word groups such as: "UG"; "bug, jug, rug, mug, plug".

I have never met such a lovable creature as *Bagasaurus* (or my subtile, "Heigh-Ho Din-O-saur"). In case you have never heard of a Bagasaurus, a short story describes that 100,000,000 years ago, when the ice age came, all the dinosaurs died off, except that one dinosaur egg (put in a blue paper bag by its parents) was saved! I am not one who ruins the great story by giving away the ending so you will just have to read the rest yourself. I will tell you that *Bagasaurus* travels around the world (in fantastic high resolution pictures) from Easter Island to Egypt and along the way he has to bag (as in *Bagasaurus*) words and images by correctly answering questions. The student uses the joystick to move and the ENTER key to collect the words. If incorrect, the Bagasaurus stomps his (I use the word "his" carefully as it may be "her" and I do not want a dinosaur to get mad at me) feet and shakes its head no. If correct he/she nods yes and takes the word or image. Sound effects are great! Each footstep, closer or further away, can be distinguished. Each nod, positive or negative, is different. Each screen provided a little different type of sound. "Heigh-Ho Din-O-saur!" There are 13 different scenes, 100 questions, 70 images, and 300 words. After correctly choosing a word or image to a question (*Bagasaurus* does not move on until you answer correctly) all chosen words are fitted into one of 50 different stories or poems. Throughout the story are underlined words and by hitting the ENTER key the student changes the story word with the word they chose with *Bagasaurus*. Vocabulary skills are developed with synonyms, antonyms, and multiple word meanings. Reading comprehension skills include classification, following directions and sequencing. Of course, there are 10 "Teaching Extensions," five activity cards, a "Bagasaurus Gameboard" and a dictionary. High-resolution graphics abound.

Something I did not mention about all these games; they are all non-resident. The disk must be connected all the time and the effect is excellent. All high-resolution pictures flow from one into the other. The music blends with the scene. Everything works perfectly.

With all I have written I bet you're wondering if I tried it out on some children. Of course, I let my children work with it. And yes, I sat back and observed their motivation and educational achievement and *yes, it does what it says!* My children, Jennifer, David and Michael loved it! They cooperated together in developing the programs, they learned the blends in *Roll-A-Word,* they made stories with *Picture Place!*, and they had *Bagasaurus* travel around the world to collect words and most important of all . . . they let me play with them.

(Radio Shack Stores nationwide, Cat. No. 26-2538, $99)

Jungle Queen
Seeks A Safari Hero

If real men don't eat quiche, it's just as unfortunate that real cannibals do eat people, and right now a group of them is adding a few vegetables, some garlic, and a little Accent to a stewpot of water rapidly heating up over a well-stoked fire. In the pot of water is our Jungle Queen, and if she is not rescued soon she will become a tasty dinner for the hungry cannibals.

But wait! In the distance a hunter on safari hears the Queen's cries for help. The area of jungle he is in is nearly impassable, so he leaps for a vine and swings from tree to tree, Tarzan-style. Suddenly the trees end and our hero encounters an alligator-infested river. He must swim across the river, avoiding and killing the alligators as he swims. He must also avoid large clusters of bubbles which can entrap him and limit his maneuverability, making him more susceptible to the hungry 'gators.

Once across the river, our hero is getting very close to the cannibal village. Some of the cannibals have discovered his presence and are rolling rocks and boulders at him as he attempts to climb the hill where they are cooking the Queen. After ducking and jumping the rolling and bouncing rocks, our intrepid rescuer arrives at the scene of the cookout where, in two mighty leaps, he jumps over the heads of the cannibal cooks and rescues his beloved Queen (clap and cheer loudly!).

Jungle Queen, by Dave Shipka of ZOSO Software, is a good rendering of the arcade game *Jungle Hunt*. It has four screens; the swinging vines, the river crossing, the rolling stones and boulders, and the rescue of the Queen in the cannibals' cook pot. In higher levels, the swinging vines have monkeys on them trying to knock the hero off the vines. While the graphics do not set a new standard for excellence as did games like *Zaxxon* and *Pyramid Adventure*, they are good. My staff of assistant reviewers (three teen-aged sons and their friends) rated this as a good game and fun to play.

When I first received this game for review, I found that the tape did not stop running when the game was loaded. I also found an oversight in the documentation — it did not specify how to move the man through the screens, whether to use the joystick or the keyboard. I wrote to Dave Shipka about the problems, and in a few days a new tape and revised documentation arrived in the mail with both problems corrected. It looks like the buyer can expect good, prompt support in case of any problem.

The program is extremely long — it requires full 32K, so if you have a disk drive, by all means order the disk version.

(ZOSO Software, 6606 Skywae Drive, Columbus, OH 43229, tape $26.95, disk $29.95 plus $2 S/H)

— **James G. Kriz**

Turn Of The Screw Correction

In Tony DiStefano's column in the December 1984 RAINBOW, the last sentence on Page 28 is incorrect. It should read: "Next, take a little piece of wire and jumper Pin 20 to pins 2, 5, 6, 9, 12, 15, 16, and 19 — one at a time." Our thanks to Rodney Masten, of Macon, Mo., for catching this.

Additionally, the BASIC listing Tony mentions as an example program was inadvertently left out. The following routine will run four lights in a chase sequence. Use the up and down arrows to increase and decrease the speed of the chase:

```
160 POKE&HFF40,1:GOSUB 250:POKE&
HFF40,2:GOSUB 250:POKE&HFF40,4:G
OSUB 250:POKE&HFF40,8:GOSUB 250:
GOTO 160
250 A$=INKEY$:IF PEEK(341)=247 T
HEN X=X+1ELSEIF PEEK(342)=247 TH
EN X=ABS(X-1)
280 FOR I=0 TO X : NEXT I:RETURN
```

The line below will turn four lights on and off randomly. Change (16) to (255) for an eight-light system.

```
380 POKE&HFF40,RND(16) : SOUND R
ND(100)+100,RND(5) : GOTO 380
```

Key Tronic Keyboard
Among The Best

Even though Radio Shack has introduced a "real" keyboard on the CoCos made over the past few months, there still seems to be quite a bit of action on the replacement keyboard front. For one thing, there are still hundreds of thousands of older models in use, and they won't go away very quickly.

Key Tronic's KB-500 is a fairly recent entry into the field. It has some good features, but also some drawbacks compared with some other keyboards. Like most of the other replacement keyboards, this one is made by one of the major keyswitch manufacturers; Key Tronic makes keyboards for many of the IBM-compatible computers and a number of other systems as well, and they are probably best known for the replacement keyboards they sell for the IBM PC and PC*jr* systems.

The Key Tronic keyboard installs easily; it's not as easy to do as the Macrotron or Radio Shack units, since you have to cut off the center post, but like them (and unlike the HJL-57 and Mark Data Super Pro) it drops right onto the existing mounting posts just like the old keyboard did. The KB-500 comes with the "F board adapter" to plug into newer CoCos' keyboard sockets, and it works on all models including the CoCo 2. The instructions are reasonably clear and should cause no problem.

The KB-500 is quite similar to many of the "IBM clone" keyboards in that its keys are the same size and shape as the PC's keys and feel much the same, but without the characteristic "Selectric click" of the IBM PC keyboard. The layout is identical to that of the normal CoCo keyboard except that one function key has been added, in contrast to the four you get on an HJL or Macrotron unit. This key acts as PF1 if you are using software designed for either of those keyboards.

One interesting aspect of the Key Tronic keyboard's design is that it seems to be geared toward the person who mostly uses programs in BASIC. The specialized shifted-key functions (caps lock, pause, delete line and brackets) are now marked on the keys themselves. (Radio Shack has been using SHIFT-0 as the caps lock key on the Model I and III as well as the CoCo for the past five years, but none of the machines has the key marked to indicate it.) Also, the BREAK and CLEAR keys — the two that will wipe out the program line you were typing — have heavier springs under them to keep them from being pressed accidentally. While this is great for BASIC, application programs will often use those keys for special purposes (*Telewriter* uses BREAK to delete characters and CLEAR for special editing functions), so you may have minor problems. It might be a good idea if Key Tronic would put two extra normal springs in the package so those who want to can restore either or both of those keys to normal and even protect other keys, if desired.

The only actual minus points I found (aside from the dull color scheme of gray alphanumeric keys and dark gray control keys) were the way the keytops stuck up from the surface, leaving a gap of about $3/16$-inch, and that Key Tronic copied the rather silly IBM practice of putting a small keytop in a large space with a "skirt" covering up the gaps (on the KB-500 the ENTER key is the only one so treated). I'd just as soon have a key that takes up the full space.

The Key Tronic KB-500 is a decent replacement for the CoCo's original keyboard, even if you have the new Radio Shack keyboard in your machine already. As I've said before, since the computer keyboard is such a personal thing, you might want to look at as many of the various replacements as you can before making your choice.

(Spectrum Projects Inc., Box 21272, Woodhaven, NY 11421, or Box 9866, San Jose, CA 95157-0866, $89.95 plus $3 S/H)

— Ed Ellers

Semigraphics Support
Combines Text And Graphics

By T. Gray

One drawback to the Color Computer is the difficulty of combining text and graphics. One way around this difficulty is *Semigraphics Support* (SGS) by Micro Computer Systems. This machine language programming utility meshes with Extended Color BASIC to provide medium-resolution graphics in nine colors plus black.

The result? You have 17 new commands and four new functions. You have easy access to single-bit and six-bit sound generation. You can use five new, full-color graphics modes, with Extended BASIC commands to draw circles, squares, and lines, fill them in, move them around, create animation, and label graphics with text.

There are a lot of good things on this disk (or tape). There are demonstration programs for 16K and 32K. These are interesting enough that I watched them through a couple of times before I even tackled the main program. The demos give you a good idea of what the program can do.

The "new" semigraphics modes are already in your CoCo, and are discussed in Section IV of *Getting Started With Color BASIC*. They provide resolutions from 32 x 16 (SG4 or alphanumeric internal, which CoCo normally uses to put

letters onscreen) to 64 x 192 (SG24). In order to access these modes, it is usually necessary to do a lot of *PEEKing* and *POKEing* around. Fortunately for me, *SGS* does not require an understanding of the hardware and its quirks.

The graphics commands are sensible, easy to remember, and similar in syntax to those of Extended BASIC. All are distinguished by a preceding 'S.' For example, SPIXL(x,y,c) sets a pixel located at (x,y) to color (c). The function A = SPIXT(x,y) is a pixel test; it returns the value (zero to eight) of the color of pixel (x,y). So much of this is familiar that—after you have spent a few hours puzzling it out—it is easy to use. There is SCIRCL for circles and ellipses. There is SLIN for lines and boxes (solid, dotted, and dashed) and for solid boxes. There is SFILL to paint, and SHIFT for easy simple animation. You can SKOPY one graphics page to another, or SMOVE graphics around the page.

The sound routines are just as handy. STONEA(f,e) gives you control over the single-bit sound routine. You can set the frequency delay (f) and the envelope delay (e). STONEB (f,e,v) gets you into the six-bit sound, where you can control the volume (v) as well. These two tones aren't as easy as PLAY but are fun to play with.

In fact, the author of the documentation seems determined to force you to play around in order to learn how to use the system. I'm not saying that the documentation is inadequate. The manual was nicely done and spiral bound. (It did what spiral bindings always do: it uncurled and dumped pages all over the floor. If looseleaf binders are too expensive to include in the package, just punch holes in the pages and I'll buy my own binder, thank you.)

No, the information is there, but the entire manual seems a bit backward. The full description of how to load

the thing into various CoCos is found at the back. At the front is an incomprehensible command summary. As I worked through the commands, I kept running into other commands and terms that had not been previously covered. This made the concepts a little hard to pick up. By the time I had worked through the manual, the command summary wasn't quite so incomprehensible, but it shouldn't be one of the first things a user trips over. It was this aspect of the manual that made the program a bit hard to get involved with.

A nice feature of the manual is that there are dozens of sample programs showing you how to use the various commands and features of the program. Even those impressive demonstration programs are listed! Not only do these help you learn the system, they are a treasure trove of information that is missing from the main sections of the text. You pass over these sample programs at your peril.

For example, you can save your semigraphics pages to cassette or disk as ML programs. This is an important feature of the system that really should be given a main section in the manual. Instead, it sneaks into a sample program somewhere toward the back few pages.

One of the main selling features of the system is the ability to easily combine text with colorful medium-resolution graphics. It delivers, too, in three styles of text (orange, green, and inverse, the same as ECB). The graphics mode for which this is available is of about the same quality as set/reset graphics, but is far easier to use. This important feature is not even mentioned in the manual. It is hidden in one of the sample programs, waiting for you to discover it. The actual method of locating letters on the screen, while not difficult, is far from clear in the sample program.

It will take you a bit of experimenting to sort everything out. This is all well and good, if you have the time. I'm sure it wasn't the author's intention to play guessing games with the user, but that's the way it comes out.

In addition to the major omissions of saving graphics and combining text with graphics, there were some minor missing pieces. I'd have appreciated having a few tables that weren't in the text:

1) summary of the commands and their syntax (the commands are summarized, but the syntax is buried in individual sections of text);
2) table of the maximum x and y values in each mode;
3) color code would be handy (it's the same as Color BASIC — 1=green, 2=yellow, etc. but I haven't memorized them);
4) since some commands are not available in some modes, a chart of this would be useful.

So there it is, a semigraphics support system that extends your Extended BASIC. Should you buy it? Well, it is easy to use once you get the hang of it. It does increase your capabilities when programming in BASIC. If you incorporate it into a program for sale, there is no license fee. It works with 16K, 32K, cassette, and disk. It comes with a pile of sample programs to help you get to know it better. It encourages, or insists on, exploration and experimentation.

Explore The Universe And Rescue The Earth In *Trekboer*

There's a deadly virus threatening the future of humankind in *Trekboer* — a new graphics Adventure by Mark Data Products — and your mission is to search the universe for a cure.

You are aboard the spaceship Trekboer as the Adventure begins, with a variety of buttons to push, doors to open, rooms to search, objects to collect, a tracking screen to watch and a robot that you will need to help you.

As in other Mark Data graphics Adventures, such as *Calixto Island, Sea Search* and *Shenanigans*, the visual effects in *Trekboer* are top-notch and of such quality that you will find yourself (again) in front of the computer during the early morning hours.

You have four planets to explore in your journey, each with its own unique environment and the inherent dangers you've come to expect from Adventure games. Be forewarned that you should be braced for some shocking discoveries because the virus has been claiming many lives.

You will need to watch your diet, too, until you discover how the disease is being transmitted. You will probably get thirsty along the way, but it's a good idea to have any liquids tested before indulging yourself.

If you find yourself in danger of being wiped out, or if it's just time to call it an evening, a SAVE feature allows you to retain the game at any point for later resumption.

There's also a HELP feature, which I found to be of questionable value because it usually repeats advice given earlier. Directional commands are simplified by allowing you to enter the first letter of the direction in which you want to proceed.

Mark Data has graciously provided maps and telecommunications aids, but first you will have to find them and, secondly, figure how they can best serve you. There is, of course, some guesswork involved, a lot of false starts, and logical steps that must be taken before you can utilize them.

You must be diligent in your search because there's no turning back to the home base unless you're willing to sacrifice your pride and/or the game. After all, the entire universe is counting upon you!

One of my first reactions was to the name of the Adventure game. Why in the world would the United States want to name a spaceship Trekboer? After thinking about it for a while, it occurs to one that the name could be a clue to the solution of the Adventure. Even then, you've got a way to go before reaching a satisfactory conclusion.

Like most Mark Data products I have experienced, *Trekboer* demonstrates the full potential of the Color Computer. The game is challenging, the options are many, the plot is sophisticated, the graphics exceptional, and the appeal is universal. If you're into Adventures, especially of the graphics variety, *Trekboer* is for you.

(Mark Data Products, 24001 Alicia Parkway, No. 207, Mission Viejo, CA 92691, tape $24.95, disk $27.95.)

— Charles Springer

Speed Racer
Offers Good Challenge
For The Racing Enthusiast

Just when you thought the Color Computer had reached its full graphics' potential, along comes a program like MichTron's *Speed Racer* that makes you wonder just how much more we can extract from this incredible machine.

As you climb into the driver's seat, you are greeted by an opening screen that includes a CoCo and monitor (with motion picture-type credits) on a red and black checkerboard field. You know then and there you are in for something special.

You have a choice of four racecourses, ranging from the predictable rectangular shape to an S-shaped track, and others with a never-ending variety of hairpin turns and tempting straightaways.

You're off with a wave of a checkered flag! Just push your gearshift (right joystick) forward and you're rapidly pushing 200 miles per hour in the age-old quest for glory. Your speed is indicated in a nicely composed speedometer at the lower left of the screen. The object of *Speed Racer* is to pass the required number of cars for each lap, which is indicated by an arrow under a 'Cars Passed' marker at the lower right. You must reach this goal to progress to the next lap. Otherwise, the game is over. Complete all five laps (10 miles) and you win the race.

The real hazards of racing, of course, are the other cars in the race. You must manage to pass them without colliding or losing control, or you will be the latest fatality. Collisions are dramatically represented on the screen by a cloud of smoke and flying tires.

The competition is represented in true arcade fashion that rivals one of those packaged in a sit-down booth at the local video gathering place. It is the game that many CoCo lovers have envied for many months, secretly hoping that it would become available to us.

The action in *Speed Racer* is not quite as furious as the one in the arcade — you won't be running through any billboads, for example — but the level of challenge would rate an eight on a scale of one to 10. There are road signs, however, which warn you of impending curves ahead. If you hug the curve for too long, you'll probably blow a tire. That's the cue for the pit crew to show its speed in replacing them, an interesting spectacle.

There are separate high-score capabilities for each track for up to 10 games. The track is selected by moving the joystick from left to right.

The scenery in *Speed Racer* is as distracting as it is beautiful, ranging from big city skylines and mountain ranges to ocean scenes, complete with sailboats. One of my favorites is the one that includes a giant suspension bridge. Some people I know play the game as much for the scenery as for the thrills of racing.

The graphics in this game are as good as I've seen on any computer, without question. Let's hope this is one of many more to come from MichTron. This effort is indicative, I hope, of an even stronger emphasis on quality in what could be a whole new era of arcade games for the Color Computer.

(MichTron, 576 S. Telegraph Road, Pontiac, MI. 48053, cassette $34.95, disk $34.95)

— **Charles Springer**

One-Liner Contest Winner . . .

This one-liner is best described as "Old Glory the easy way." It's a surprisingly good (for one line) reproduction of the Stars and Stripes.

Ronald W. Wilson
Franklin, OH

The listing:

```
1 PMODE1:PCLS:SCREEN1,0:DRAW"BM0
,0R255D99L255U99BR50D50L50BU40BR
50R205D10L205D10R205D10L205D10R2
55D10L255D10R255D10L255D10R255":
FORX=7T042STEP11:FORY=9T044STEP1
2:CIRCLE(X,Y),4,4:NEXTY:NEXTX:PA
INT(3,3),3,4:FORA=9T089STEP20:PA
INT(52,A),4,4:NEXTA:GOTO1
```

(For this winning one-liner contest entry, the author has been sent copies of both The Rainbow Book Of Adventure *and its companion* Rainbow Adventure Tape.*)*

CORRECTIONS

"The ABC Game" (September 1984, Page 157): James F. Taylor advises us that this program will not run as is in 16K. He suggests deleting the following remark lines:

1-10, 145-185, 275-305, 411-415, 461-464, 481-483, 561-565, 591-596, 681-685, 841-845, 911-915, 1001-1009

"The Trip" (August 1984, Page 90): Jacques Bourgeois tells us that a reader questioned his use of a minus sign in place of an equal sign in the statement *IF RND (2)- 1 THEN . . .* in Line 222. He assures us that this unusual syntax does work properly in this instance.

"Pretty As A Picture" (May 1984, Page 137): Tom Szlucha states that there are a number of problems with the program, and sends along a corrected listing of the first 47 lines of the program. Here it is:

```
0 CLEAR 500:AR=0:GOTO220:'DUMPFI
X 9/18/84
10 AR=1:CX=0:PI=INT(PX/S+.5):FOR
X=0 TO255 STEPPX:P=S*-X:F=0:B=0:
FORY=191TO0STEP-1
20 IFPPOINT(X,Y)=CL THEN 40 ELSE
 80
30 GOTO 80
40 IF F>0 THEN 60
50 IF B>0 THEN GOSUB 160
60 F=F+S:GOTO 100
70 IF B> 0 THEN 90
80 IF F>0 THEN GOSUB 170
90 B=B+S
100 NEXT Y
110 IF B>0 THEN GOSUB 160
120 IF F> 0 THEN GOSUB 170
130 PRINT#-2,"M"BD","P
140 NEXTX
150 GOTO 180
160 PRINT#-2,"R"+STR$(B)+","+STR
$(CX):B=0:RETURN
170 PRINT#-2,"J"+STR$(F)+","+STR
$(CX):F=0:RETURN
180 CLS:INPUT"ANOTHER COLOR Y/N"
;R$:IF R$="Y" THEN AR=1:GOTO290
190 PRINT#-2,"M0,-100":PRINT#-2,
"A"
200 POKE 150,87:POKE65494,0
210 CLS:PRINT"FINISHED":END
220 CLS:PRINT"CGP-115 SCREEN DUM
P":PRINT:INPUT"DRAW TEST PATTERN
 Y/N";R$
230 IF R$="Y" THEN GOSUB 470
240 PS=PEEK(&HB6)
250 PMODE PS
260 CLS:INPUT"PLOTTING SCALE 1X
OR 2X";S:IF S<1 OR S>2 THEN 260
270 S=INT(S)
280 IF PS=3 THEN PX=2 ELSE PX=1
290 CLS
300 IF PS=3 THEN PRINT"PMODE 3 C
```

```
OLORS                    0=BLACK
                         1=GREEN
       5=BUFF             2=YELLOW
       6=CYAN             3=BLUE
       7=MAGENTA          4=RED
       8=ORANGE"
310 IF PS=4 THEN PRINT"PMODE 4 C
OLORS                    0=BLACK
       5=BUFF             1=GREEN"
320 PRINT:PRINT"BE SURE YOU INPU
T COLOR WHICH IS COMPATIBLE WITH
 SCREEN MODE""
330 PRINT:INPUT"COLOR TO SENSE";
CL:IF CL<0 OR CL>8 THEN 330
340 CLS:PRINT"0=BLACK
1=BLUE           2=GREEN
3=RED"
350 PRINT:INPUT"DRAW WITH WHICH
COLOR PEN";CP:IF CP<0 OR CP>3 TH
EN 350
360 IF AR=1 THEN PRINT#-2,"H":GO
TO430
370 BD=(480-192*S)/2:CX=0
380 POKE 65495,1
390 POKE 150,180
400 PRINT#-2,CHR$(18)
410 PRINT#-2,"M0,-300*S"
420 PRINT#-2,"I"
430 PRINT#-2,"C"CP
440 PRINT#-2,"M"BD","STR$(CX)
450 SCREEN1,0:PMODE PS,1
460 GOTO10
```

This should be typed in after deleting everything before Line 470.

"Cooking With CoCo" (November 1984, Page 146): Colin Stearman passes along a note from a reader who says that Radio Shack's disk versions of *EDTASM, Scripsit* and *Spectaculator* (all of which use the Color TRSDOS operating system) use bytes 17 and 18 of the directory entry, which conflicts with the file creation date area used by his program. Dates are not put into the file when these programs create files and they should pose no real problem. Colin also cautions that files created by these programs should not be redated with the BASIC program on Page 148. This problem may also occur with future Radio Shack programs that use Color TRSDOS (which is supplied on each program disk in each package).

"Adding Auto Answer To Modem I" (November 1984, Page 19): Tony Sharp advises us that there may be a problem with installing his modification in the newer Modem IB, due to some changes in the circuit board. One difference that has been noted is that the voltage regulator is now called VR1 instead of U1.

"CC-Talk" (November 1984, Page 50): There was an error in production that misplaced part of two lines. Here they are in the correct form:

```
140 CLS:PRINTTAB(8)"BUFFER CONTE
NTS":PRINTTAB(6)"spaceMORE/enter
EXIT":PRINT
170 BB=TB:BE=&H7CFF' buffer star
t and end addresses
```

Finalizing The Rainbow Checkbook

By Richard White
Rainbow Contributing Editor

T his is the "Beginners Issue" where we welcome all of you that have recently become proud owners of a CoCo and have joined the RAINBOW family of readers. We will mark the occasion with a bit of a celebration, for *Rainbow Checkbook, Version 1.0* is complete, and the listing follows this column.

Over the past few months, I have been writing about the program as it developed. Both program logic and syntax have been discussed in their natural setting, a working program. One consequence has been the need to revise the code previously discussed and explain the reasons for the changes. Changes come about for two reasons. First, for a better idea. As the programmer works with a program, inadequacies become evident and revisions to fix these are installed. Second are the pure bugs.

Even with extensive testing, obscure bugs continue to be found in most programs. That's why I say *Version 1.0*

---

is the first to be published and not the first one written. It is reasonably clean and should do most of the job that needs to be done. It is hardly a final version if such a version exists. Since a program continues to evolve with time, it should be organized to facilitate change. To me, that means defining specific program functions and assigning blocks of line numbers to these functions. I made a table of these functions at the start and have stayed with it throughout.

"Version 1.0 . . . is hardly a final version . . . Since a program continues to evolve with time, it should be organized to facilitate change."

LINES	FUNCTION
1-9	Most Frequently used subroutines
10-99	Other subroutines
100-199	Enter items
200-299	Edit records and update balances
400-499	Strip cleared items to make updated current file
600-699	Print report
900-999	Tape and disk I/O
2000-2099	Initialization
2100-2199	Make new file
10000	Save program routine
10100	PCLEAR 1 routine

Version 1.0 contains all the functions shown in the table in the line number blocks shown. Note that the most frequently used lines go to the beginning of the program and the least used lines to the end. Each time there is a *GOSUB* or *GOTO*, BASIC starts at the beginning of the program to search for the required line. It makes no sense for it to have to continually search over lines that are used only once during the normal operation of the program. So, all initialization goes to the end rather than the beginning of the program where you will see it many times. On the other hand, frequently called subroutines should be at the beginning where BASIC can quickly find them. And, the smaller the line number, the fewer bytes it takes to store the subroutine call. In any case, I like small line numbers. They are easier to remember. I also see no need to keep increments of 10 between line numbers as you can see from the listing.

Another word of advice to the beginner. If you organize your program from the beginning, you will not need to renumber it. *RENUM* will destroy all the structure you carefully developed

and make trouble-shooting much more difficult. There is a place for *RENUM* — it is useful to make a user version of a program, numbered at increments of one. This both shortens the program and speeds it a tad. But, keep your structured source version for editing and upgrading. Renumbering the listing to increments of one saves 84 bytes in spite of the fact that I had already reduced memory usage by putting the subroutines at the beginning.

A good program today should be as self-documenting as possible. Still, some words of wisdom are generally needed to get the user off on the right foot. Perhaps the best place to start is to explain why the program was written in the first place. You see, there was this pile of bank statements on my desk — about six months' worth. Reconciling the checkbook to the bank statements is a pain and the program I wrote way back when did not do that. I looked at the old program, but it was written before I discovered structuring a BASIC program. Starting from scratch looked easier. The real celebration came yesterday when I worked down through the (now nine months') pile, finding less than $100 worth of mistakes. It's a relief to know that there will be no OM (out of money) messages from the bank.

The easiest way for you to get up and running with *Rainbow Checkbook* is to buy a Jan. 1985 RAINBOW ON TAPE. As published, the program runs in a 16K tape-based Extended BASIC machine. Those with disk drives should have 32/64K machines or be willing to cut some of the code out. For example, you might make two versions, one without the printer block starting at 600 and one without the editor and New-month routines starting at 200 and 400. This way you can enter and edit data with one version, save the file, and load the other version to print the file.

If you have been typing the program in with me over the last few months, check your code against the listing. There have been a number of changes, both added features and debugs.

When you load and run the program, you are shown the first menu that offers lOAD FILE or nEW FILE options. Choose new file by keying 'N'. You are now prompted for account information including name, bank, account number and date. You are then asked for your last bank statement balance. This is what the bank thinks your balance is and not what is shown in your checkbook. Since there are always some

outstanding items in your checkbook that the bank does not know about, the bank's balance must be different from any shown in your checkbook. Now you enter all the outstanding items in your checkbook. Your options are shown in the lines at the bottom of the entry screen. You can scroll back and forth through your file using the up and down arrows and edit entries. However, you cannot add items or make changes from the review mode.

"A correction entry allows you to adjust your balance without changing the calculated bank balance. It's a way to fudge your balance to match what the bank says you have."

When all items are entered, exit by keying 'S' to get to I/O routines. Here you can save to tape, disk or printer. If you do print your file, you will be returned to the normal input screen where you can key 'S' again to save your file to tape or disk. I use the name CURRENT and let BASIC append the default disk extension DAT. The next time you enter the program choose 'L' to load your file.

The input screen is nearly the same as New-file except a *CLEAR* function has been added. When you get a bank statement, first make sure that all items in your checkbook are in your current file. Next, go through your checkbook, check off all items that have cleared the bank and correct any entry errors. Now, load your current file and step back through it using the up-arrow key. Compare each entry in the file with that in your checkbook. If an item has been cleared, key 'C' and the calculated bank balance will be adjusted, then the status letter will change from 'O' to 'C'. If you clear an item by mistake, key 'C' again to toggle back to 'O' status. If you find mistakes in the file, key 'E' to get to edit mode to make the correction.

Warning! Never edit a cleared item. The program does not adjust the bank balance for changes made in the edit mode. If you need to edit a cleared item, first toggle it back to 'O', edit it and then toggle the status back to 'C' from input.

In edit mode you can change the check number, key 'N' then the new number, change an entry to adjustment or deposit, key 'A' or 'D', or change an adjustment to a correction. For a correction key 'C'. Correction does not show on the command line since it was a very late (read yesterday) addition. A correction entry allows you to adjust your balance without changing the calculated bank balance. It's a way to fudge your balance to match what the bank says you have. Note that you cannot enter a correction directly from the input mode. Rather, enter an adjustment and edit the entry to the correction type.

When you enter a check, enter it as a positive number and the program automatically deducts the amount from your checkbook balance. A deposit is automatically added to the balance. An adjustment may be either positive or negative. An interest credit should be entered as a positive number since it increases your balance. Service charges should be entered as negative numbers.

In input mode, you can duplicate the previous entry, like the date, simply by hitting ENTER for that field. The same happens in edit, so you only key in entries for fields you need to change. When you leave edit, you are returned to input, but at the end of the file. Already I see a number of changes for *Version 2.0.*

When you have corrected your file and cleared all items that have cleared the bank for the month, your bank balance from the program should match that shown on the statement. When it doesn't (to err is human, it's not the computer's fault), look for errors both in your current file in the program and for errors in your checkbook. A printout is very handy here. Key 'S' to get to the Save Menu, then 'P' for the Printer. Change the Baud rate, if necessary, by hitting ENTER and then keying in the rate you want. Next, use the down-arrow to move the cursor next to PRINT and press ENTER. The program will print the account information, allow you to update the date if necessary and then list the current file. When all is right, print a copy of the current file to store with the checks and bank statement for the month.

Finally, you will want to use the New-month routine to strip all cleared items out of the current file to a tape or disk archive file. This routine saves a file of cleared items that you might title JAN85 or JAN85/CLR for disk. It also

saves the full current file with the extension BAK if you are on disk. If you loaded the file from tape, New-month saves to tape. The same applies with disk. Finally, New-month strips all cleared items from your current file and returns you to input. At this point you should save this new current file.

Newcomers to RAINBOW should know that this month's "Bits and Bytes of BASIC" is different from normal in that I did not discuss any BASIC code. Next month we will resume our line by line discussion of the program. I will take up the printer code first and show some do's and don'ts in making BASIC print a formatted, multi-page report. In the meantime, do some testing of *Rainbow Checkbook*. Suggestions for improvements to go into *Version 2.0* may be sent to me care of THE RAINBOW.

15	92	600	65
32	131	640	100
68	134	950	2
200	180	2110	79
232	231	END	46
410	208		

The listing:

```
0 GOTO10100
1 PRINTCHR$(191);:I$=INKEY$:IFI$
="""THENFORX=0TO5:NEXT:PRINTCHR$(
8);:GOTO1ELSEPRINTCHR$(8);:RETUR
N
2 F=INSTR(F1,A$(V),"$"):RETURN
```

```
4 PRINT:PRINT"current date: ";DA
$
5 LINEINPUTI$:IFI$<>""THENDA$=I$
:RETURNELSERETURN
10 LC$=CC$:LD$=CD$:LA=CA:LB=CB:L
S$=CS$:LN$=CN$
11 PRINT@96,USINGS4$;LC$;:PRINT"
"LD$:PRINT@108,USINGSS$;LA:PRI
NT@119,USINGSS$;LB;:PRINTSTRING$
(63,32);:PRINT@128,USINGSN$;LS$;
:PRINTLN$:RETURN
12 F1=1:V=CR-1:GOSUB2:LC$=MID$(A
$(V),F1,F-F1):F1=F+1:GOSUB2:LD$=
MID$(A$(V),F1,F-F1):LS$=MID$(A$(
V),F+1,1):LN$=MID$(A$(V),F+3,LEN
(A$(V))-F+3):LA=A(V,0):LB=A(V,1)
:RETURN
14 CC$=LC$:CD$=LD$:CA=LA:CB=LB:C
S$=LS$:CN$=LN$:IFF1=2THENRETURN
15 PRINT@PS,USINGS4$;CC$;:PRINT"
"CD$:PRINT@PS+12,USINGSS$;CA:P
RINT@PS+23,USINGSS$;CB;:PRINTSTR
ING$(63,32);:PRINT@PS+32,USINGSN
$;CS$;:PRINTCN$:RETURN
16 F1=1:V=CR:GOSUB2:CC$=MID$(A$(
V),F1,F-F1):F1=F+1:GOSUB2:CD$=MI
D$(A$(V),F1,F-F1):CS$=MID$(A$(V)
,F+1,1):CN$=MID$(A$(V),F+3,LEN(A
$(V))-F+3):CA=A(V,0):CB=A(V,1):R
ETURN
17 IFCR>1THENCR=CR-1:GOSUB14:GOS
UB12:GOSUB11:RETURNELSERETURN
18 IFCR<LR-1THENCR=CR+1:GOSUB10:
GOSUB16:GOSUB15:RETURNELSEGOSUB1
0:PRINT@224,STRING$(95,32):FG=0:
IFCR<LR THENCR=CR+1:RETURNELSERE
TURN
22 IFI$=CHR$(13)THENCC=CC+1:CC$=
STR$(CC):     CC$=RIGHT$(CC$,LEN(C
C$)-1):PRINT@224,USINGS4$;CC$:RE
TURN
24 IFVI>0THENPRINTUSING"#";VI;:L
INEINPUTI$:CC$=STR$(VI)+I$:CC$=R
IGHT$(CC$,LEN(CC$)-1):CC=VAL(CC$
):RETURN
26 CC$="DEP":PRINT@224,CC$;:RETU
RN
28 CC$="ADJ":PRINT@224,CC$;:RETU
RN
29 CC$="CORR":PRINT@224,CC$;:RET
URN
```
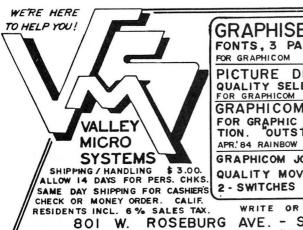
```
30 PRINT@229,"";:LINEINPUTI$:IFI
$<>""THENCD$=I$
31 PRINT@229,USING"%    %";CD$
32 PRINT@236,"";:IFFG<>1THENLINE
INPUTI$:I=VAL(I$):IFI<>ØTHENCA=I
 ELSEELSECA=Ø
34 PRINT@236,USINGSS$;CA;:RETURN
36 IFVAL(CC$)=ØTHENCB=CB+CA ELSE
CB=CB-CA
37 PRINT@247,USINGSS$;CB;:PRINTU
SINGSN$;CS$;:LINEINPUTI$:IFI$<>"
"THENCN$=I$ELSECN$=" "
38 A$(CR)=CC$+"$"+CD$+"$"+LEFT$(
CS$,1)+"$"+CN$:A(CR,Ø)=CA:A(CR,1
)=CB:RETURN
40 CR=CR+1:LR=CR:GOSUB10:PRINT@2
24,STRING$(95,32):FG=Ø:RETURN
50 PRINT@8,RC$,"ITEM    DATE     A
MOUNT    BALANCE":RETURN
52 PRINT@352,"LAST BANK BALANCE
";:PRINTUSINGSS$;BB:RETURN
54 PRINT@384,"<ENTER> =NEXT CHEC
K #","1ST DIGIT STARTS NEW # SER
IES":RETURN
56 PRINT@448,"dEPOSIT aDJUSTMENT
 vOID cLEARED eDIT sAVE lOAD nEW
 MONTH "CHR$(94)"'S";:RETURN
58 PRINT@448,"dEPOSIT aDJUSTMENT
 vOID","eDIT sAVE"CHR$(94)"'S";:
RETURN
60 GOSUB22:GOSUB30:GOSUB36:GOSUB
40:RETURN
62 GOSUB26:GOSUB30:GOSUB36:GOSUB
40:RETURN
64 GOSUB28:GOSUB30:GOSUB36:GOSUB
40:RETURN
66 I$=CHR$(13):GOSUB22:FG=1:GOSU
B30:GOSUB36:FG=Ø:GOSUB40:RETURN

68 PRINT@320,"REVIEW ENTRIES
    ";:PO=335:FG=1:GOSUB17:RETU
RN
100 CLS:PS=224:GOSUB50:GOSUB52:G
OSUB54:GOSUB56:PRINT@128,STRING$
(192,32):CR=LR:GOSUB12:GOSUB11:F
1=2:GOSUB14:F1=Ø:CC=VAL(CC$)
110 PRINT@320,"ENTER OUTSTANDING
 ITEMS            ";:PO=224
120 PRINT@PO,"";:GOSUB1:VI=VAL(I
$):IFFG=ØAND(VI>ØOR I$=CHR$(13))
GOSUB60:GOTO120
130 J=INSTR(1,"DAVCESLN"+CHR$(94
)+CHR$(10),I$):IFFG=ØTHENONJ GOS
UB62,64,66,160,200,900,950,400,6
8,18ELSEONJ GOSUB2160,2160,2160,
160,200,900,950,400,17,18
140 IFJ>4 ANDJ<9THEN100
150 IFCR=LR THEN110ELSE120
160 IFCS$="O"THENCS$="C":GOSUB15
:GOSUB38:IFLEFT$(CC$,4)<>"CORR"T
HENIFVAL(CC$)=ØTHENBB=BB+CA ELSE
BB=BB-CA ELSEELSE170
165 PRINT@371,"";:PRINTUSINGSS$;
BB;:PRINT@343,"";:RETURN
170 CS$="O":GOSUB15:GOSUB38:IFLE
FT$(CC$,4)<>"CORR"THENIFVAL(CC$)
=ØTHENBB=BB-CA ELSEBB=BB+CA
175 PRINT@371,"";:PRINTUSINGSS$;
BB;:PRINT@343,"";:RETURN
200 CLS:GOSUB50:PRINT@72,"UNEDIT
ED ENTRY":PRINT@201,"EDITED ENTR
Y":PRINT@321,"TYPE FIELD ENTRY C
ORRECTION"," <enter> FOR NEXT FI
ELD":IFFG=1THENTF=1:FG=Ø
210 I$=INKEY$:PS=96:GOSUB16:GOSU
B15:PRINT@224,STRING$(95,32);:PR
INT@416,"nEW #, dEP, aDJ, -=DELE
TE, +=ADDUP/DOWN ARROWS, rETURN
    ";:PS=224:CT$=CC$:TA=CA
215 GOSUB1:IFI$=CHR$(13)THENPRIN
T@224,CC$;:GOTO240ELSEJ=INSTR(1,
"NDA-+R"+CHR$(94)+CHR$(10),I$):O
NJ GOSUB220,26,28,225,230,217,23
5,237
216 IFFG=1THENFG=Ø:GOTO210ELSEIF
I$<>"R"THEN:GOTO240ELSEFG=TF
```

```
217 RETURN
220 PRINT@224,"";:GOSUB1:VI=VAL(
I$):IFI$="A"THENGOSUB28ELSEIFI$=
"C"THENGOSUB29ELSEGOSUB22
221 RETURN
225 FORX=CR TOLR-2:A$(X)=A$(X+1)
:A(X,0)=A(X+1,0):IFVAL(CC$)>0THE
NA(X,1)=A(X+1,1)+CA:CB=CB+CA ELS
EA(X,1)=A(X+1,1)-CA:CB=CB-CA
227 NEXT:PRINT@96,STRING$(95,32)
;:CC$="":CD$="":CA=0:CS$="":CN$
="ITEM DELETED":CR=CR-1:LR=LR-1:
FG=1:RETURN
230 CC$=" ":CD$=" ":CS$=" ":CA=0
:TA=0:CN$="MAKE NEW ENTRY":FG=1:
FORX=LR TOCR+1STEP-1:A$(X)=A$(X-
1):A(X,0)=A(X-1,0):A(X,1)=A(X-1,
1):NEXT:CB=A(X-1,1):LR=LR+1:GOSU
B38
232 PS=96:GOSUB16:GOSUB15:PRINT@
416,"nEW #, dEP, aDJ, -=DELETE":
PRINTSTRING$(28,32);:PS=224:GOSU
B1:FG=0:J=INSTR(1,"NDA-",I$):ONJ
GOSUB220,26,28,225
234 IFJ>0THEN240ELSE232
235 FG=1:IFCR>1THENCR=CR-1:GOSUB
16:PS=96:GOSUB15:PS=224:RETURNEL
SERETURN
237 FG=1:IFCR<LR-1THENCR=CR+1:GO
SUB16:PS=96:GOSUB15:PS=224:RETUR
NELSERETURN
240 PRINT@416,STRING$(45,32):GOS
UB30:IFVAL(CT$)=0THENIFVAL(CC$)=
0THENTA=CA-TA:GOTO242ELSETA=-TA-
CA:GOTO242
241 IFVAL(CC$)=0THENTA=TA+CA ELS
ETA=TA-CA
242 PRINT"TA"TA,"CA"CA;:FORX=1TO
2000:NEXT: FORX=CR TOLR-1:A(X,1)
=A(X,1)+TA:NEXT:CB=A(CR,1):PRINT
@247,USINGSS$;CB;:LINEINPUTI$:IF
I$<>""THENCS$=I$
244 PRINT@256,USINGSN$;CS$;:LINE
INPUTI$:IFI$<>""THENCN$=I$
246 PRINT@259,CN$:GOSUB38:GOTO21
0
400 CLS:PRINT@8,RC$,"   STRIP &
SAVE CLEARED ITEMS",,,,,TAB(8)"t
APE",,,,TAB(8)"dISK",,:GOSUB1:IF
I$="D"THEND=1ELSEIFI$="T"THEND=-
1ELSE400
410 PRINT:PRINT:IFD=-1THENPRINT"
  TAPE FILE"ELSEPRINT"  DISK FIL
E"
415 PRINT:PRINT"FILENAME";:IFD=1
THENPRINT" AND EXTENSION"
420 LINEINPUTCF$:GOSUB4:CT=0:FOR
CR=1TOLR-1:GOSUB16:IFCS$="C"THEN
CT=CT+1:NEXTELSENEXT
```

```
425 OPEN"O",D,CF$:PRINT#D,CT+1:P
RINT#D,BB:PRINT#D,CB:PRINT#D,AN$
:PRINT#D,BA$:PRINT#D,AC$:PRINT#D
,DA$:FORCR=1TOLR-1
430 GOSUB16:IFCS$="C"THENPRINT#D
,A$(CR):PRINT#D,A(CR,0):PRINT#D,
A(CR,1):NEXTELSENEXT
435 CLOSE:CT=0:OPEN"O",D,NM$+"/B
AK":PRINT#D,LR:PRINT#D,BB:PRINT#
D,CB:PRINT#D,AN$:PRINT#D,BA$:PRI
NT#D,AC$:PRINT#D,DA$:FORX=1TOLR-
1:PRINT#D,A$(X):PRINT#D,A(X,0):P
RINT#D,A(X,1):NEXT:CLOSE
440 CLOSE:CT=0:FORCR=1TOLR-1:GOS
UB16:IFCS$="O"THENCT=CT+1:A$(CT)
=A$(CR):A(CT,0)=A(CR,0):A(CT,1)=
A(CR,1):NEXTELSENEXT
445 FORX=CT+1TOLR-1:A$(X)="":NEX
T:LR=CT+1:CR=CT:RETURN
600 'TARGET LINE
605 BP=PEEK(150):BU=INSTR(1,"16*
18**41******87*****************180"
,RIGHT$(STR$(BP),LEN(STR$(BP))-1
)):BU=9600/BU
605 POKE65494,0:BP=PEEK(150):BU=
INSTR(1,"16*18**41******87******
*********180",RIGHT$(STR$(BP),LEN
(STR$(BP))-1)):BU=9600/BU
610 CLS:PRINT@137,"PRINT ROUTINE
S",,," CURRENT BAUD RATE ="BU
,TAB(6)"RESET BAUD RATE",,,"
PRINT",,,," RETURN":CT=1
615 CT$=RIGHT$(STR$(CT),1):LP=IN
STR(1,"  1 2 3",CT$):LP=LP*3
2+2:PRINT@LP,">";:GOSUB1:PRINT@L
P," ";:IFI$=CHR$(10)ANDCT<3THENC
T=CTT+1:GOTO615ELSEIFI$=CHR$(94)
ANDCT>1THENCT=CT-1:GOTO615ELSEIF
I$<>CHR$(13)THEN615ELSEONCT GOTO
620,630,100
620 PRINT" ";:INPUT"ENTER NEW BA
UD RATE";BU$:BU$=LEFT$(BU$,1):BL
=INSTR(1,"361249",BU$):IFBL=0THE
NPRINT"baud rate error":SOUND100
,50:GOTO605
625 BU(1)=180:BU(2)=87:BU(3)=41:
BU(4)=18:BU(5)=6:BU(6)=1:BU=BU(B
L):POKE150,BU:GOTO605
630 CLS:PRINT@96,"SET PRINTER AN
D PAPER AND PRESS any key.":GOSU
B1:FORX=1TO3:PRINT#-2," ":NEXT:P
RINT#-2,TAB(8)"ACCOUNT NAME:    "
AN$:PRINT#-2,TAB(8)"BANK NAME:
   "BA$:PRINT#-2,TAB(8)"ACCOUNT
NUMBER: "AC$
640 GOSUB4:PRINT#-2,TAB(8) "DATE
PRINTED:      "DA$:PRINT#-2," ":P
RINT#-2,TAB(8)"CALCULATED BALANC
E AT BANK ="BB:PRINT#-2," ":PRIN
```

```
T#-2,TAB(8)"CHECK# DATE       AMOU
NT    BALANCE    STATUS NOTE":PRINT
#-2," ":PG=13
650 FORCR=1TOLR-1:GOSUB16:PRINT#
-2,TAB(9)"";:PRINT#-2,USING"%   %
";CC$;:PRINT#-2,USING"  %      %";C
D$;:PRINT#-2,TAB(22)"";:PRINT#-2
,USINGSS$;CA;:PRINT#-2,TAB(32)""
;:PRINT#-2,USINGSS$;CB;:PRINT#-2
,"   "CS$"       "CN$
655 PG=PG+1:IFPG>61THENFORX=1TO7
:PRINT#-2," ":NEXT:PRINT#-2,TAB(
8)"CHECK# DATE   AMOUNT    BALANCE
    STATUS NOTE":PRINT#-2," ":PG=5
660 NEXT:FORX=PG TO67:PRINT#-2,"
":NEXT:RETURN
900 CLS:PRINT@8,RC$,"        SAVE
OR PRINT DATA",,,,TAB(8)"tAPE",
,,,TAB(8)"dISK",,,,TAB(8)"pRINTE
R":GOSUB1:IFI$="D"THEND=1ELSEIFI
$="T"THEND=-1ELSEIFI$="P"THEN600
ELSE900
905 GOSUB4
910 GOSUB985:OPEN"O",D,NM$:PRINT
#D,LR:PRINT#D,BB:PRINT#D,CB:PRIN
T#D,AN$:PRINT#D,BA$:PRINT#D,AC$:
PRINT#D,DA$:FORX=1TOLR-1:PRINT#D
,A$(X):PRINT#D,A(X,0):PRINT#D,A(
X,1):NEXT:CLOSE:RETURN
950 CLS:PRINT@8,RC$,TAB(11)"LOAD
 DATA",,,,,TAB(8)"tAPE",,,,TAB(8
)"dISK":GOSUB1:IFI$="D"THEND=1EL
SEIFI$="T"THEND=-1ELSE950
960 GOSUB985:OPEN"I",D,NM$:INPUT
#D,LR,BB,CB:LINEINPUT#D,AN$:LINE
INPUT#D,BA$:LINEINPUT#D,AC$:LINE
INPUT#D,DA$:GOSUB970:FORX=1TOLR-
1:INPUT#D,A$(X):INPUT#D,A(X,0):I
NPUT#D,A(X,1):NEXT:CLOSE
962 CR=1:GOSUB16:IFVAL(CC$)=0THE
NA(0,1)=A(1,1)-A(1,0)ELSEA(0,1)=
A(1,1)+A(1,0)
965 PRINT:PRINT"PRESS ANY KEY TO
 CONTINUE ":GOSUB1:RETURN
970 CLS:PRINT@98,"FILENAME= "NM$
:PRINT:PRINT"  ACCOUNT INFORMATI
ON":PRINT:PRINT"NAME: "AN$:PRINT
"BANK: "BA$:PRINT"ACCT #: "AC$:P
RINT"FILE UPDATED: "DA$:RETURN
985 PRINT:IFD=-1THENPRINT"TAPE F
ILE"ELSEPRINT"DISK FILE"
990 PRINT:PRINT"FILENAME";:IFD=1
THENPRINT" AND EXTENSION: ",NM$E
LSEPRINT": "NM$
995 LINEINPUTI$:IFI$<>""THENNM$=
I$:RETURNELSERETURN
2000 CLEAR3500:DIMA$(100),A(100,
1)
2050 RC$="RAINBOW CHECKBOOK":SS$
="##,###.##":SN$="% %":CR=1:LR=1
:S4$="%  %":CS$="O":A$(0)="  $ $
$FIRST ENTRY BELOW"
2100 CLS:PRINT@2,"rainbow checkb
ook version 1.0":PRINT@47,"BY":P
RINT@73,"RICHARD WHITE":PRINTTAB
(7)"COPYRIGHT (C) 1984":PRINT:PR
INT:PRINTTAB(8)"lOAD FILE":PRINT
:PRINTTAB(8)"nEW FILE"
2110 GOSUB1:IFI$="L"GOSUB950:GOT
O100ELSEIFI$<>"N"THEN2110
2115 CLS:PRINT@96," enter accoun
t name":LINEINPUTAN$:PRINT:PRINT
" enter bank name":LINEINPUTBA$:
PRINT:PRINT" enter account #":LI
NEINPUTAC$
2120 CLS:GOSUB50:PRINT:PRINT:PRI
NT"ENTER LAST BANK STATEMENT
   BALANCE     ";:LINEINPUTI$:BB=VA
L(I$):PRINT@128,STRING$(64,32):P
RINT@128,"note:  REPORTED CHECKB
OOK         BALANCE WILL NOT MATCH
    YOUR          BALANCE UNTIL OUTSTAND
ING"
2125 PRINT"CHECKS, CHARGES, DEPO
SITS  AND    CREDITS AT STATEMENT
  TIME ARE     ENTERED.";:GOSUB1:CB
=BB:GOSUB52:GOSUB54:GOSUB56:PRIN
T@128,STRING$(192,32)
2130 PRINT@320,"ENTER OUTSTANDIN
G ITEMS            ";: PO=224:PS=PO
2140 PRINT@PO,"";:GOSUB1:VI=VAL(
I$):IFFG=0AND(VI>0OR I$=CHR$(13)
)GOSUB60:GOTO2140
2150 J=INSTR(1,"DAVES"+CHR$(94)+
CHR$(10),I$):IFFG=0THENONJ GOSUB
62,64,66,200,900,68,18ELSEONJ GO
SUB2160,2160,2160,200,900,17,18
2155 IFJ=5 THENGOTO100
2157 IFCR=LR THEN2130ELSE2140
2160 RETURN
10000 SAVE"RNBWCKBK
10100 PCLEAR1:GOTO2000
20000 FORX=0TO64:PRINT@X,CHR$(32
+X):PRINT@X+96,CHR$(PEEK(1024+X)
);:NEXT
```

The 6809 Instruction Set — Zeroing In On 'The Real Thing'

By R. Bartly Betts
Rainbow Contributing Editor
With Programs By Chris Bone

I am sure you are finger-weary if you typed in all of last month's listing by hand. You deserve a treat and I hope we will have one for you this month.

I think we are almost through the painful part and about ready to have fun. There is still a lot of work to be done, but the work is a lot sweeter when something worthwhile is being accomplished. With the information you now have learned about registers, we can begin on the real thing, the 6809 instruction set.

You have already been using several of the instructions recognized by your assembler. This month you will begin taking a closer look at these instructions, what they do and how to use them.

A great start toward understanding your assembler's instructions is knowing how to interpret the instruction set reference in your assembler manual and the references in other 6809 assembly language programming books. The

(Bartly Betts is a former reporter, magazine editor and store owner now residing in Fort Worth, Texas. He has owned and operated a Color Computer for over three years and is presently taking an electronics and computer course by correspondence. Chris Bone is a college computer science major and has been programming for more than three years. He averages between six and nine hours a day on the CoCo.)

EDTASM+ manual seems like a good place to begin. I have the one that came with my cartridge pack, Radio Shack Cat. No. 26-3250. The instruction set references start on Page 39. Page 37 has a section on definition of terms and Page 38 explains the abbreviations used in the set. It will be helpful to refer to these pages when you run up against definition problems. You are probably most familiar with the LD (load) instruction (pages 44 and 45 in my manual). Regardless of what assembler you use, the reference should look something like the following:

```
LD
Load Register from Memory
(8-Bit)
   Source Forms: LDA P; LDB P
   Operation: R'<--M
   Condition Codes:
      H -Not affected
      N -Set if the loaded data is
         negative; cleared otherwise
      Z -Set if the loaded data is zero;
         cleared otherwise
      V -Always cleared
      C -Not affected
   Description: Loads the contents of
memory location M into the designated
register.
```

Addressing Modes: Immediate; Extended; Direct; Indexed.

If you take a close look at this instruction, it will provide a basis from which to understand the rest of the instruction set.

First, notice that the reference I typed in is one of two for the LD instruction. This one is labeled as "8-BIT." This instruction is only 8-bit because it only uses registers A and B, both 8-bit registers. You will recall from your study of binary numbers that the maximum value an 8-bit number can contain is 255. If you want to handle numbers larger than 255, you will have to use the 16-bit form of the LD instruction. Actually, the only difference between the two forms of the load instruction is that the 8-bit instruction uses registers A and B while the 16-bit form uses registers X, Y, S and U.

The information about what registers are used in what way is found in the line "Source Forms." The italic 'P' in the line represents a memory location, such as:
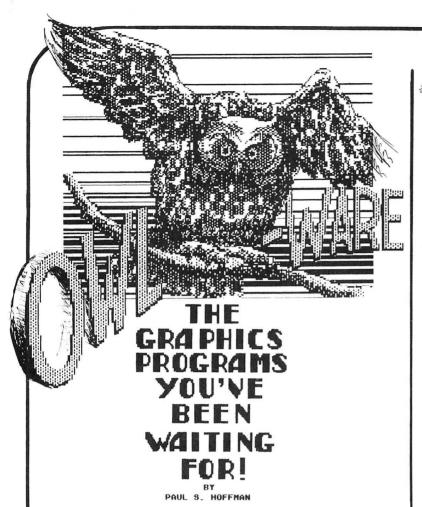
```
LDA 32
```
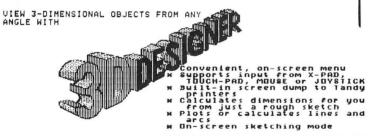
where A is the register to be loaded and 'P' is memory location 32.

The line beginning with "Operation:" tells you that a register after the LD operation (R'), has been loaded with a value from memory (M). The arrow in the line indicates the direction of the data flow.

The lines following "Condition Codes:" indicate the effect that the operation has on the five condition code flags. At this point you need not worry about them, their function will become clear in later columns.

The "Description:" line restates the function of the operation. In this case it tells you that a register (either A or B) is loaded with the contents of a memory location (indicated by P).

The last line, "Addressing Modes:" deserves a large portion of this month's column. Keeping the addressing modes straight is not difficult once you have had a chance to investigate how they work, so let's investigate the possible modes.

Immediate Addressing

In immediate addressing, the data for an instruction is the actual value of the instruction's operand. An example is:

LDA #32

The # symbol tells the assembler that the value of 32 decimal is to be loaded into register A. A hexadecimal value could be indicated by:

LDA #$20

The $ symbol tells the assembler that the value to be loaded is hexadecimal. As before, the # symbol causes a load of the actual value of Hex 20 or decimal 32.

Extended Addressing

It is just as easy to load the contents of a memory location into a register as it is to load an established value. Extended addressing can be used to cause the value of a memory location to be transferred to a register. This is done by leaving off the # symbol from the instruction, such as:

LDB 32

In this case, register B is loaded with the value that is contained in memory location 32. Again, you can follow the

same procedure for Hex:

LDB $20

If you glance through some of the other instructions in your assembler manual you will see that a great many instructions can operate in both the immediate and extended addressing modes.

Indexed Addressing

Registers X and Y are known as index registers, as noted last month. Although they are not used exclusively for indexed addressing, it is their main function. As 16-bit registers, they can manipulate values up to $FFFF or 65536, which includes all of your computer's memory. In indexed addressing, you load registers X or Y with values that serve as pointers to a memory location. For example:

00100 START	LDX	#$500	*Load X with a memory value
00110	LDA	#$20	*Load Hex 20 into A
00120	STA	,X	*Store the value of A into the memory location pointed to by register X

In this example, register X is used as a pointer to memory location $500. You use a comma before the X to show that it is being used as an index to point to a location in memory. The power of indexed addressing lies in the ease with which the "pointer" in the X and Y registers can be adjusted, for example:

| 00100 START | LDX | #$500 | *Load X with A memory value |
| 00110 | LDA | #$20 | *Load A with Hex 20 |

00120	LDB	#$21	*Load B with Hex 21
00130	STA	,X+	*Store A in memory location $500
00140	STB	,X	*Store B in memory location $501

A plus symbol after the register X indicates that it is to be incremented after A is loaded into the memory address to which it is pointing. Thus the STB (store B) command stores the contents of register B into the next memory location, 500+1. An operation such as this could be put into a loop and any number of values could be loaded into sequential memory locations.

If you have not yet done so, it might be better to look at the ST instruction in your assembler manual. You will notice that its purpose is to transfer the value in a register to a memory location, as we have indicated above. The ST instruction can be either an 8- or 16-bit operation and it can operate in extended, direct or indexed addressing.

Inherent Addressing

With this addressing mode we will take a break from mental activity. It is the easiest of all addressing modes to understand because it does not require anything more than knowing what the instructions are and when to use it. Inherent addressing is exactly what its name implies, it provides all of its own information, nothing more needs to be added. For example:

RTS

is an instruction that causes program control to be returned to the parent routine from a subroutine. No address or values are needed nor, indeed, can

be included. The return address for the instruction is automatically pulled from the S or hardware stack. There are a number of instructions that use inherent addressing. You might wish to glance through the instruction references and pick out some of the others, like CLR and SWI.

Direct Page Addressing

Direct page or direct addressing involves using the contents of the direct page register as the most significant byte of an address. In effect, your assembler does this anyway, but, on start up, the direct page register contains 0 and thus has no effect on other addresses. Once you put a value into the DP register, however, it is used as an offset for all 8-bit addressing.

In order to load the DP register, you must use the TFR (transfer) instruction, it cannot be loaded directly. For example:

LDB #$44 *Load the B register with the value of $44

TFR B,DP *Transfer the contents of B (44) to DP

will load a value of 44 into the direct page register. Now, if you issue an instruction:

LDB $20

the assembler will use the value of $44 as the most significant byte of the address whose contents were loaded into B. The effective address will be $4420. As already mentioned, 8-bits can only hold a maximum value of 255. Thus a direct page address can be any of 256 bytes (0 through 255), beginning at the direct page register address.

Now, after saying all that, I will tell you that we probably won't be doing anything with direct page addressing in the near future, so if it seems confusing, don't worry. You could live quite a happy life without ever using this type of addressing.

Now For The Good Stuff

That's all there is to the various types of addressing. With that under your belt it is time to quit the chatter for this column and get on with the real thing.

Listing 1 at the end of the column was written by Chris to make use of some of the information we have passed on this month. Try and figure out what it does before reading the comments. If you can guess what is going on, you have been making great progress. Even if you only recognize some of what the program is for, you should be pleased with yourself. Take the time to study the program thoroughly, it isn't too long, but it contains several good routines that will be.

If all is not clear after studying the program, perhaps the followng will help: the crux of the program is that a green snake is created on a black screen with a green border. You can control the snake with a joystick plugged into the right joystick port. If the snake hits the screen border it is stopped and must change directions to continue. Chris has put lots of comments into the program to help you follow it through. Again, look it over carefully. Try and pick out routines you can load and assemble and try out on your own to see how they work and to see if you can change them to do something a little

TCE News Release

In 1985 TCE Will Introduce
CHILD'S PLAY

Mouse Technological Software For The Color Computer!

Ted Malaska, President & Co-founder of TCE Programs Inc., announced today a new division for the development of business software.

The new division will distribute a series of machine language business

programs, under the name *Child's Play* beginning in the first half of 1985. The *Child's Play* series will use mouse technology and what Mr. Malaska termed *"Floating Overlays"*, to control the operations within the program.

When asked why the name *Child's Play* was selected for a business series, Mr. Malaska responded: "Floating Overlays will make the program operation seem like *Child's Play*, compared to other business software of today."

different; it is the way you will learn the quickest.

The Test

Now for the hard part. To see if you are really getting the hang of assembly language code, see if you can create a second snake to slither around with the first one, controlled with the left joystick. Use Chris' code to guide you in your efforts. Chris and I would be pleased to see the result of your efforts.

The Second Installment

If that isn't enough to keep you occupied during the next month, here is the second part of the 51-column text generator. This month we are including the portion of the code that will translate keystrokes into machine language generated characters. You will be able to use the new text screen after typing and and assembling the following source code. Before typing in this month's code, however, you will need to load last month's listing and renumber it according to Chris' instructions given with the listing. You will also need to type in and *RUN* the accompanying BASIC driver. The steps you should follow are:

1) Type in this month's source code.
2) When finished, append last month's source code to the new code.
3) Delete the *ORG* line from the beginning of last month's code.
4) Renumber the combined codes. Check your assembler manual for how to do this.
5) Save the combined source codes.
6) Assemble the combined codes.
7) If there are any mistakes, correct them and assemble the code again.
8) Save the resulting machine language code.
9) Type in and save the BASIC listing (Listing 3).
10) When you are sure there are no mistakes in either the machine language code or the BASIC listing, load the machine language listing into memory. Then load the BASIC listing into memory.
11) Run the BASIC listing. You should be presented with a new, 51-column screen and the prompt, "Type what you want:"
12) Begin typing, as fast as you like.

13) Enjoy the new screen. It will soon be yours to use any time you like. It will also soon have several other programming features.

You may find it useful to compare this month's source code with last month's BASIC listing. As the two serve the same purpose, comparing them will help you understand what the source code listing is accomplishing and how it does the job.

The characters you see on your screen are now the text to be used in the finished program and will be 51 columns across the screen. Still to be installed are the hooks to disable the usual text mode and put you entirely into the new mode. As well, some of the other programming features that will accompany the new screen mode are not yet included.

Listing 2 is the source code for the new portion of the 51-column screen program. Listing 3 is the BASIC driver you will need to use the new 51-character screen. Next month we will do away with all BASIC help and you will be in full 51 columns.

Listing 1:

```
          00010 **************************
          00020 *NO ORG LINE IS GIVEN    *
          00030 *ASSEMBLE THIS PROGRAM   *
          00040 *INTO MEMORY WHEREVER    *
          00050 *EDTASM+ PUTS IT AND RUN*
          00060 *IT FROM EDTASM+ WITH    *
          00070 *GSTART IN Z-BUG         *
          00080 *DELETE REMARKS TO SAVE  *
          00090 *RAM ON 16K SYSTEMS. IF*
          00100 *YOU ARE USING THE DISK  *
          00110 *EDTASM+ THEN USE THE    *
          00120 *MEMORY EFFICIENT WAY AS*
          00130 *OUTLINED IN THE MANUAL *
          00140 *THESE GUIDELINES APPLY  *
          00150 *FOR ALL PROGRAMS IN THE*
          00160 *BYTEMASTER BOTH PAST    *
          00170 *AND FUTURE AND I AM     *
          00180 *SORRY FOR ANY PROBLEMS *
          00190 *THAT HAVE OCCURED.      *
          00200 *              CHRIS     *
          00210 **************************
0000 C6 80     00220 START  LDB  #$80      *CHR$(128)=BLACK BLOCK
0002 BD A92A   00230        JSR  $A92A     *CALL BASIC'S CLS ROUTINE
          00240 **************************
          00250 *DRAW BORDER             *
          00260 **************************
0005 4F        00270        CLRA           *SRART X COORD @ 0
0006 5F        00280 LOOP   CLRB           *SET Y COORD @ 0
0007 17 0082   00290        LBSR SET       *SET (A,B)
000A C6 1F     00300        LDB  #$1F      *SET Y COORD @ 31
000C 8D 7E     00310        BSR  SET       *SET (A,B)
000E 4C        00320        INCA           *X COORD = X COORD + 1
000F 81 40     00330        CMPA #$40      *ALL THE WAY TO THE RIGHT?
0011 26 F3     00340        BNE  LOOP      *IF NOT SET THE NEXT PAIR OF POINTS
0013 4F        00350 LOOP1  CLRA           *SET X COORD @ 0
0014 8D 76     00360        BSR  SET       *SET (A,B)
0016 86 3F     00370        LDA  #$3F      *SET X COORD @ 63
0018 8D 72     00380        BSR  SET       *SET (A,B)
001A 5A        00390        DECB           *Y COORD = Y COORD - 1
001B 2A F6     00400        BPL  LOOP1     *IF Y COORD >0 THEN SET MORE POINTS
001D 8E 00B1   00410        LDX  #SNAKE    *POINT TO BEGINNING OF TABLE OF VALUES
0020 108E 000A 00420        LDY  #10       *10 POINTS IN THE SNAKE
0024 CC 2010   00430        LDD  #$2010    *A=32 B=16 (CENTER OF SCREEN)
0027 ED 81     00440 SCLR   STD  ,X++      *SAVE INTO TABLE
0029 31 3F     00450        LEAY -1,Y      *Y=Y-1
002B 26 FA     00460        BNE  SCLR      *IF WE HAVEN'T INITIALIZED THE WHOLE TABLE LOOP
002D 8D FC     00470 WAIT   LDD  SNAKE     *GET FIRST POINT IS SNAKE (BACK END)
002F 8D 6E     00480        BSR  RESET     *SET IT TO BLACK
0031 8E 00B3   00490        LDX  #SNAKE+2  *GET START OF POINT LIST TO SET
0035 EC 84     00500 WLP    LDD  ,X        *GET FIRST POINT
0037 ED 1E     00510        STD  -2,X      *SAVE IT AS NEXT TO LAST ELEMENT
0039 34 10     00520        PSHS X         *SAVE CURRENT POSITION IN SNAKE
003B 8D 4F     00530        BSR  SET       *SET THE POINT
003D 35 10     00540        PULS X         *GET BACK CURRENT SNAKE POS
003F 30 02     00550        LEAX 2,X       *POINT TO NEXT ELEMENT
0041 8C 00C5   00560        CMPX #SNAKE+20 *ALL DONE?
0044 26 EF     00570        BNE  WLP       *IF NOT LOOP
0046 EC 1C     00580        LDD  -4,X      *GET LAST ELEMENT SET
0048 ED 1E     00590        STD  -2,X      *SAVE AS NEW LAST ELEMENT
004A 34 10     00600        PSHS X         *SAVE PLACE IN TABLE
004C AD 9F A00A 00610       JSR  [$A00A]   *CALL ROM JOYSTICK ROUTINE
0050 EC 84     00620        LDD  ,Y        *GET JOYSTK(0) INTO A,JOYSTK(1) INTO B
0052 35 10     00630        PULS X         *RETREIVE TABLE POINTER
0054 81 20     00640        CMPA #32       *STICK LEFT OR RIGHT?
0056 22 08     00650        BHI  NOLEF     *IF RIGHT THEN GOTO NOLEF
0058 6A 1E     00660        DEC  -2,X      *X COORD = X COORD - 1
005A 26 0E     00670        BNE  UPDN      *IF IT ISN'T ZERO THEN WE ARE STILL ON SCREEN
005C 6C 1E     00680        INC  -2,X      *OTHERWISE MOVE IT BACK
005E 20 0A     00690        BRA  UPDN      *GO CHECK UP/DOWN
0060 6C 1E     00700 NOLEF  INC  -2,X      *X COORD=X COORD + 1
0062 A6 1E     00710        LDA  -2,X      *GET X COORD
0064 81 3E     00720        CMPA #$3E      *ALL THE WAY RIGHT?
0066 23 02     00730        BLS  UPDN      *IF NOT GO CHECK UP/DOWN
0068 6A 1E     00740        DEC  -2,X      *OTHERWISE MOVE IT BACK
006A C1 20     00750 UPDN   CMPB #32       *STICK UP OR DOWN?
006C 22 08     00760        BHI  NOUP      *B>32 MEANS DOWN
006E 6A 1F     00770        DEC  -1,X      *Y COORD=Y COORD - 1
0070 26 08     00780        BNE  PAUSE     *IF STILL ON SCREEN THEN GO PAUSE
0072 6C 1F     00790        INC  -1,X      *MOVE BACK ON SCREEN
0074 20 0A     00800        BRA  PAUSE     *GO PAUSE
0076 6C 1F     00810 NOUP   INC  -1,X      *Y COORD= Y COORD + 1
0078 E6 1F     00820        LDB  -1,X      *GET Y COORD
007A C1 1E     00830        CMPB #$1E      *STILL ON SCREEN?
007C 23 02     00840        BLS  PAUSE     *IF SO GO PAUSE
007E 6A 1F     00850        DEC  -1,X      *OTHERWISE MOVE IT BACK
0080 8E 5000   00860 PAUSE  LDX  #$5000    *SET DELAY LEN (1=SHORT FFFF=LONG)
0083 BD A7D3   00870        JSR  $A7D3     *CALL ROM DELAY ROUTINE
0086 BD A1C1   00880        JSR  $A1C1     *CHECK KEYBOARD
0089 27 A2     00890        BEQ  WAIT      *IF NO KEY KEEP GOING
008B 39        00900        RTS            *BACK TO BASIC
          00910 **************************
          00920 *SET POINT @ A,B         *
          00930 *USE ROM ROUTINE TO      *
          00940 *CALCULATE MOST OF THE   *
          00950 *STUFF                   *
          00960 **************************
008C 34 06     00970 SET    PSHS A,B       *SAVE A,B FOR RETURN TO CALLER
008E 8E 0098   00980        LDX  #SETRET   *GET READY TO TRICK ROM
0091 34 10     00990        PSHS X         *SAVE SETRET AS RETURN ADDRESS
0093 34 02     01000        PSHS A         *STACK X COORD
0095 7E A8D7   01010        JMP  $A8D7     *GTO ROM ROUTINE
0098 A6 84     01020 SETRET LDA  ,X        *WE ARE BACK!!!! X=SCREEN LOC TO WORK ON
009A 9A 86     01030        ORA  ,S        *SET BIT INDICATED BY ROM ROUTINE
009C A7 84     01040        STA  ,X        *SAVE BACK ON SCREEN
009E 35 86     01050        PULS A,B,PC    *GET X AND Y COORDS BACK AND QUIT SET ROUTINE
          01060 **************************
          01070 *RESET ROUTINE, WORKS    *
          01080 *ALMOST THE SAME AS SET  *
          01090 **************************
00A0 34 06     01100 RESET  PSHS A,B       *SAVE A AND B
00A2 8E 00AC   01110        LDX  #RSRET    *RETURN ADDRESS
00A5 34 10     01120        PSHS X         *SAVE RETURN ADDRESS
00A7 34 02     01130        PSHS A         *SAVE X COORD
00A9 7E A8D7   01140        JMP  $A8D7     *CALL ROM ROUTINE
00AC BD A8B5   01150 RSRET  JSR  $A8B5     *CALL SECOND ROM ROUTINE TO RESET
00AF 35 86     01160        PULS A,B,PC    *WASN'T THAT EASY!
00B1          01170 SNAKE  RMB  20
     0000     01180        END
```

Listing 2:

```
          00010 **************************
          00020 *51 BY 24 COLUMN DRIVER  *
          00030 *TO BE CALLED FROM BASIC *
          00040 *WITH A USR FUNCTION     *
          00050 **************************
7000          00060        ORG  $7000
7000 4D       00070 START  TSTA           *NUMBER OR STRING IN USR?
7001 27 13    00080        BEQ  QUIT      *IF NUMBER QUIT
7003 E6 84    00090        LDB  ,X        *B=LENGTH OF PASSED STRING
7005 10AE 02  00100        LDY  2,X       *Y=START ADDRESS OF PASSED STRING
7008 5C       00110        INCB           *LEN=LEN+1 (CUE QUIT ON NO LENGTH)
7009 5A       00120 OLP    DECB           *LEN=LEN-1
700A 27 0D    00130        BEQ  QUIT      *IF DONE QUIT
700C A6 A0    00140        LDA  ,Y+       *GET FIRST CHR INTO A:Y POINTS TO NEXT CHR
700E 34 24    00150        PSHS B,Y       *SAVE LENGTH AND LOC
7010 8D 05    00160        BSR  GO        *GO PRINT CHR$(A)
7012 35 24    00170        PULS B,Y       *GET BACK LENGTH AND LOC
7014 20 F3    00180        BRA  OLP       *OUTPUT NEXT CHR
7016 39       00190 QUIT   RTS            *RETURN TO BASIC
          00200 **************************
          00210 *PRINT CHR$(A)           *
          00220 **************************
7017 81 08    00230 GO     CMPA #8        *BACK SPACE?
7019 26 0B    00240        BNE  NOTBS     *IF NOT BRANCH
701B 17 00E0  00250        LBSR BACK      *MOVE CURSOR BACK 1
701E 86 20    00260        LDA  #32       *GET ASC FOR SPACE (BLANK OUT)
7020 8D 3A    00270        BSR  LETTER    *PRINT SPACE
7022 17 00D9  00280        LBSR BACK      *BACK IT UP AGAIN
7025 39       00290        RTS            *ALL DONE BACKSPACE
7026 81 0C    00300 NOTBS  CMPA #12       *CLEAR SCREEN?
7028 26 12    00310        BNE  NOTCL     *BRANCH IF NOT
702A 9E BA    00320        LDX  $BA       *START OF CURRENT HIRES PAGE
702C BF 712A  00330        STX  POS       *SAVE AS CURRENT CURSOR LOCATION
702F 7F 7128  00340        CLR  BIT       *CLEAR BIT WITHIN BYTE FLAG
7032 CC FFFF  00350 CLS    LDD  #$FFFF    *D=SOLID WHITE
7035 ED 81    00360 CLL00  STD  ,X++      *SAVE TO SCREEN
7037 9C B7    00370        CMPX $B7       *AT END OF CURRENT HIRES SCREEN?
7039 26 FA    00380        BNE  CLL00     *IF NOT GO AGAIN
703B 39       00390        RTS            *ALL DONE CLEAR SCREEN
703C 8D 1E    00400 NOTCL  BSR  LETTER    *SEND CHR$(A) TO SCREEN
703E 9E 712A  00410        LDX  POS       *GET CURRENT CURSOR POS
7041 9C B7    00420        CMPX $B7       *AT END OF HIRES SCREEN
7043 25 16    00430        BLO  RET       *NO SCROLL NEEDED THEN
7045 30 89 FF00 00440      LEAX -256,X    *BACK UP POINTER ONE LINE
7049 BF 712A  00450        STX  POS       *SAVE AS NEW CURSOR LOC
704C 9E BA    00460        LDX  $BA       *GET START OF CURRENT HIRES SCREEN
```

```
704E EC 89 0100  00470 SCLOOP LDD    256,X  *GET BYTE FROM LINE BELOW WHEREVER WE ARE
7052 ED 81       00480         STD    ,X++   *SAVE IN NEW LINE
7054 BC 712A     00490         CMPX   POS    *MOVED ALL LINES?
7057 26 F5       00500         BNE    SCLOOP *IF NOT GO AGAIN
7059 20 D7       00510         BRA    CLS    *CLEAR LAST LINE OF SCREEN (PARTIAL CLS)
705B 39          00520 RET     RTS           *RETURN TO BASIC
705C 81 0D       00530 LETTER  CMPA   #$0D   *IS THE CHR A CARRIAGE RETURN
705E 26 0A       00540         BNE    LETT2  *IF IT ISN'T JUST OUTPUT IT
7060 7F 712B     00550         CLR    POS+1  *SET BYTE WITHIN LINE TO ZERO(LHS)
7063 7C 712A     00560         INC    POS    *POINT CURSOR TO NEXT LINE
7066 7F 712B     00570         CLR    BIT    *SET BIT WITHIN BYTE TO LHS
7069 39          00580         RTS           *RETURN
706A C6 04       00590 LETT2   LDB    #4     *4 BYTES / LETTER IN TABLE
706C F7 712C     00600         STB    BYTE   *SAVE THE NUMBER 4(NUMBER OF BYTES LEFT)
706F 80 20       00610         SUBA   #$20   *A=A-32 (ANY CONTROL CODES HAVE BEEN DONE)
7071 3D          00620         MUL           *FIND ENTRY ON TABLE
7072 C3 712D     00630         ADDD   #TABLE *ADD TABLE OFFSET
7075 1F 02       00640         TFR    D,Y    *PUT PUT TABLE POINTER INTO Y
7077 BE 712A     00650         LDX    POS    *GET CURRENT CURSOR LOC
707A A6 A0       00660 LOOP    LDA    ,Y+    *GET BYTE FROM TABLE
707C 34 02       00670         PSHS   A      *SAVE THE BYTE
707E 8A 0F       00680         ORA    #$0F   *MAKE LSN ALL ONES
7080 8D 16       00690         BSR    SLICE  *PRINT THIS SLICE TO SCREEN
7082 35 02       00700         PULS   A      *GET BYTE BACK
7084 48          00710         ASLA          *MOVE
7085 48          00720         ASLA   *      LSN
7086 48          00730         ASLA   *      INTO
7087 48          00740         ASLA   *      MSN
7088 8A 0F       00750         ORA    #$0F   *MAKE LSN ALL ONES
708A 8D 0C       00760         BSR    SLICE  *PRINT THIS SLICE
708C 7A 712C     00770         DEC    BYTE   *NEXT BYTE TO BE DONE
708F 26 E9       00780         BNE    LOOP   *IF MORE BYTES THEN LOOP
7091 30 89 FF00  00790         LEAX   -256,X *POINT BACK TO TOP OF LETTER
7095 8D 40       00800         BSR    FORW   *MOVE CURSOR FORWARD ONE POSITION
7097 39          00810         RTS           *RETURN
7098 F6 7128     00820 SLICE   LDB    BIT    *GET CURRENT BIT WITHIN BYTE
709B 5C          00830         INCB          *ADD ONE TO IT
709C F7 7129     00840         STB    BIT+1  *SAVE AS TEMPORARY BIT COUNTER

709F 34 02       00850         PSHS   A      *SAVE SLICE
70A1 CC F800     00860         LDD    #$F800 *READY TO BUILD 5 BIT MASK
70A4 FD 7126     00870         STD    MASK   *SAVE AS CURRENT MASK
70A7 35 02       00880         PULS   A      *GET BACK SLICE
70A9 C6 FF       00890         LDB    #$FF   *SOLID WHITE
70AB 7A 7129     00900 BITTST  DEC    BIT+1  *BIT COUNT = BIT COUNT - 1
70AE 27 0E       00910         BEQ    SAVE   *IF ALL DONE THEN THE MASK IS READY
70B0 1A 01       00920         ORCC   #1     *SET CARRY FLAG
70B2 46          00930         RORA          *MOVE SLICE
70B3 56          00940         RORB   *      INTO 2 BYTES
70B4 1C FE       00950         ANDCC  #$FE   *CLEAR CARRY FLAG
70B6 76 7126     00960         ROR    MASK   *MOVE MASK TO MATCH
70B9 76 7127     00970         ROR    MASK+1 *   IT IS 2 BYTES LONG
70BC 20 ED       00980         BRA    BITTST *GO SEE IF MORE MOVEMENT IS NEEDED
70BE 34 06       00990 SAVE    PSHS   D      *SAVE SLICE(NOW USES TWO BYTES)
70C0 FC 7126     01000         LDD    MASK   *GET WORKING MASK
70C3 AA 84       01010         ORA    ,X     *WHITE OUT SCREEN SECTION
70C5 EA 01       01020         ORB    1,X    *   (IT CAN SPAN TWO BYTES)
70C7 A7 84       01030         STA    ,X     *SAVE MASK BACK TO SCREEN
70C9 E7 01       01040         STB    1,X    *   (ONCE AGAIN IT IS TWO BYTES)
70CB 35 06       01050         PULS   D      *GET BACK WORKING MASK
70CD A4 84       01060         ANDA   ,X     *SET REQUIRED BITS
70CF E4 01       01070         ANDB   1,X    *IN BOTH BYTES
70D1 ED 84       01080         STD    ,X     *SAVE TO SCREEN
70D3 30 88 20    01090         LEAX   32,X   *MOVE TO NEXT LINE
70D6 39          01100         RTS           *ALL DONE THIS SLICE
                 01110 ****************************
                 01120 *MOVE CURSOR FORWARD 5  *
                 01130 *BITS (ONE CHARACTER)   *
                 01140 ****************************
70D7 F6 7128     01150 FORW    LDB    BIT    *GET CURRENT BIT WITHIN BYTE
70DA CB 05       01160         ADDB   #5     *PLUS 5 BITS
70DC C1 07       01170         CMPB   #7     *IF MORE THAN SEVEN THEN CHANGE BYTE
70DE 23 04       01180         BLS    BITSV  *ELSE WE ARE DONE
70E0 30 01       01190         LEAX   1,X    *ADD ONE TO SCREEN LOC
70E2 C0 08       01200         SUBB   #8     *RESET BIT COUNT
70E4 C1 07       01210 BITSV   CMPB   #7     *VERY RIGHT EDGE?
70E6 26 0F       01220         BNE    BITDN1 *IF NOT THEN NO NEW LINE
```

```
70E8 1E    1Ø    Ø1230        EXG    X,D      *PUT X WHERE WE CAN WORK ON IT
70EA C1    1F    Ø1240        CMPB   #$1F     *VERY RIGHT OF SCREEN
70EC 26    Ø7    Ø1250        BNE    BITDN    *IF NOT THEN SAVE CURRENT POS
70EE 4C          Ø1260        INCA            *ELSE NEXT LINE
70EF 5F          Ø1270        CLRB            *AND RESET BYTE WITHIN LINE
70FØ 1E    Ø1    Ø1280        EXG    D,X      *SWAP D AND X BACK
70F2 5F          Ø1290        CLRB            *BIT WITHIN BYTE=Ø
70F3 2Ø    Ø2    Ø1300        BRA    BITDN1   *GO SAVE NEW POS
70F5 1E    Ø1    Ø1310 BITDN  EXG    D,X      *SWAP D AND X BACK
70F7 F7  7128    Ø1320 BITDN1 STB    BIT      *SAVE BIT WITHIN BYTE
70FA BF  712A    Ø1330        STX    POS      *SAVE CURRENT CURSOR LOC
70FD 39          Ø1340        RTS             *ALL DONE MOVING FORWARD
                 Ø1350 *************************
                 Ø1360 *MOVE CURSOR BACK 5 BITS*
                 Ø1370 *************************
70FE 34    Ø4    Ø1380 BACK   PSHS   B        *SAVE B FOR CALLER
71ØØ F6  7128    Ø1390        LDB    BIT      *GET CURRENT BIT WITHIN BYTE
71Ø3 BE  712A    Ø1400        LDX    POS      *GET CURRENT CURSOR POS
71Ø6 CØ    Ø5    Ø1410        SUBB   #5       *BIT=BIT - 5
71Ø8 2A    14    Ø1420        BPL    BACKSP   *IF STILL OK THEN SAVE IT
71ØA CB    Ø8    Ø1430        ADDB   #8       *ELSE FIX BIT COUNT
71ØC 3Ø    1F    Ø1440        LEAX   -1,X     *AND BACK UP CURSOR POS
71ØE 1E    1Ø    Ø1450        EXG    X,D      *SWAP 'EM AGAIN
711Ø C1    FF    Ø1460        CMPB   #$FF     *NEED TO MOVE TO NEW LINE?
7112 26    Ø8    Ø1470        BNE    BACKS1   *IF NOT SAVE NEW POS
7114 C6    1F    Ø1480        LDB    #$1F     *END OF LINE
7116 1E    1Ø    Ø1490        EXG    X,D      *SWAP 'EM BACK
7118 C6    Ø2    Ø1500        LDB    #2       *BIT WITHIN BYTE=2
711A 2Ø    Ø2    Ø1510        BRA    BACKSP   *SAVE
711C 1E    1Ø    Ø1520 BACKS1 EXG    X,D      *SWAP 'EM YET AGAIN
711E BF  712A    Ø1530 BACKSP STX    POS      *SAVE CURRENT POSITION
7121 F7  7128    Ø1540        STB    BIT      *SAVE BIT WITHIN BYTE
7124 35    84    Ø1550        PULS   B,PC     *RETURN
7126             Ø1560 MASK   RMB    2        *MASK FOR STORING TO SCREEN
7128             Ø1570 BIT    RMB    2        *BIT WITHIN BYTE
712A ØEØØ        Ø1580 POS    FDB    $E00     *START OF SCREEN ($600) FOR NON-DISK
712C             Ø1590 BYTE   RMB    1        *SLICES LEFT

712D FFFF        Ø16ØØ TABLE  FDB    $FFFF    *DATA TABLE FROM LAST MONTH
                 Ø161Ø ***************************
                 Ø162Ø *DELETE THE ORG LINE    *
                 Ø163Ø *FROM LAST MONTH'S PART *
                 Ø164Ø *OF THIS PROGRAM AND RE-*
                 Ø165Ø *NUMBER IT TO START AT  *
                 Ø166Ø *16ØØ, THEN TYPE IN THE *
                 Ø167Ø *LINES LISTED HERE UP TO*
                 Ø168Ø *LINE 159Ø              *
                 Ø169Ø *ASSEMBLE TO PROGRAM TO *
                 Ø17ØØ *DISK OR TAPE           *
                 Ø171Ø *USE $7ØØØ IN LINE 6Ø   *
                 Ø172Ø *FOR 32K MACHINES OR    *
                 Ø173Ø *$3ØØØ FOR 16K MACHINES *
                 Ø174Ø *USE THE NUMBER FROM    *
                 Ø175Ø *LINE 6Ø IN LINE 2Ø OF  *
                 Ø176Ø *THE BASIC PROGRAM AND  *
                 Ø177Ø *THE NUMBER-1 IN LINE 1Ø*
                 Ø178Ø *OF THE BASIC PROGRAM   *
                 Ø179Ø ***************************
```

Listing 3:

```
10 CLEAR1,&H6FFF:CLEAR300
20 DEFUSR0=&H7000
30 PMODE4,1:SCREEN1,1
40 A$=CHR$(12)+"Type what you wa
nt !"+CHR$(13)
50 GOSUB 90
60 A$=INKEY$:IF A$=""THEN60
70 GOSUB90:GOTO60
80 GOTO80
90 A$=USR0(A$):RETURN
```

CoCo Community

We have compiled a list of Color Computer Clubs because of the many requests we have received. CoCo Clubs may wish to exchange newsletters, share ideas for topics of discussion at monthly meetings, etc.

Please let us know if we have omitted any clubs and send us complete up-to-date addresses. Only those clubs which have signed our "agreement form" will appear in this listing of CoCo Clubs. Also, please notify us if you wish to add or delete any names on this list. Send your information to:

CoCo Clubs
THE RAINBOW
9529 U.S. Highway 42
P.O. Box 385
Prospect, KY 40059

ALABAMA
Birmingham Alabama Color Computer Club, Robert Matthews, 1725 14th Ave. S., Birmingham, 35205, (205) 933-0887

Huntsville Color-80 Users Group, Randy Niemann, 1011 Versailles Dr., S.E., Huntsville, 35803, (205) 882-2485

ALASKA
Alaska Color Computer Users' Group, Rick McDannel, 430C Beluga Ave., Ft. Richardson, 99505, (907) 428-0392

ARIZONA
Huachuca Hot Pokers, James Standley, 235-A Jeffords Street (NBU-96H), Ft. Huachuca, 85613, (602) 458-8338

Tucson Color Computer Club, William H. Nunn, 6857 A Lighting Circle #22, Tucson, 85708

ARKANSAS
Central Arkansas CoCo Club, Melinda Braslovsky, 1203 Erving Rdg., LP Cabot, 72023, (501) 982-8854

CALIFORNIA
Color America Users Group, Mark Randall, 2227 Canyon Rd., Arcadia, 91006, (213) 355-6111

The Davis CoCoNuts, Adam Sherman, 1818 Haussler Dr., Davis, 95616, (916) 758-3195

Sacramento Color Computer Club, Wayne Chrisope, P.O. Box 9, Elverta, 95626

Joe Bennett, 1169 Florida Street, Imperial Beach, 92023, (474-6213)

Los Angeles-Wilshire Color Computer Users' Group,c/o Norm Wolfe, 269 S. Lafayette Park Pl., Los Angeles, 90057

Ventura County Color Computer Club (VC4), c/o Pete Lydall, Oxnard Public Library, 214 "C" Street, Oxnard, 93030, (805) 984-1842 or DATA (805) 984-1842

South Bay Color Computer Club, Robert L. Cakebread, P.O. Box 3540, Redondo Beach, 90277, (213) 376-1169

Sacramento CoCo Club, Charles Doughty, Box 60511, Sacramento, 95860

Salinas Color Computer Club, Larry Livingston, 501 Monterey Highway, Salinas, 93908, (408) 484-9697

Citrus Color Computer Club, Joseph Kohn, P.O. Box 6991, San Bernadino, 92412, (714) 792-8721

Local Color—A CoCo Club of San Francisco, Andrew G. Kieval, P.O. Box 421242, San Francisco, 94142

San Joaquin CoCo Club, Steven Paul Moreno, P.O. Box 99024, Stockton, 95209, (209) 951-3938

Silicon Valley Color Computer Club, Shawn Jipp, P.O. Box 61593, Sunnyvale, 94088, (408) 749-1947

CoCo Nutz Computer Club, Walter V. Seay, 68-461 Highway 86, Thermal, 92274, (619) 397-4252

COLORADO
Lowry Microcomputer Club/CoCo Users' Group, Jerry D. Surritte, 2249 Moline St., Aurora, 80010, (303) 343-3273

Colorado Color Computer Club, Joe Applegate, P.O. Box 33492, Northglenn, 80233, (303) 650-9768

FLORIDA
Northwest Florida CoCo Nuts, William N. Lamb, P.O. Box 1032, Fort Walton Beach, 32549, (904) 244-5281

Jacksonville Color Computer Club, William H. Brown III, 2411 Hirsch Ave., Jacksonville, 32216, (904) 721-0282

CoCo Chips Color Computer Club, 6 Belle Mead Circle, Largo, 33540, (813) 581-7779

Broward County Color Computer Club, Timothy D. Neary, 510 S.W. 64th Ave., Margate, 33068, (305) 972-4074

C.C. Club of Sarasota, Ernie Bontrager, 4047 Bee Ridge Rd., Sarasota, 33582, (813)921-7510

Broward CoCo Club, Sue Spahn, 11950 N.W. 29th Manor, Sunrise, 33582, (305) 741-4737

ILLINOIS
Glenside Color Computer Club, Chuck Roberg, 521 Canyon, Carol Stream, 60188, (312) 690-9374

Town & Country CoCo, 5461 S. Kenwood, Chicago, (312) 493-3748

Cook County Color Computer Club, Tim Powers, 1633 Fifth Ave., Chicago Heights, 60411, (312) 747-7062

Illinois Color Computer Club of Elgin, Steven Stroud, 1098 Florimond Dr., Elgin, 60120, (312) 695-3186

Northern Illinois Color Computer Club, 580 Milton Lane, Hoffman Estates, 60194, (312) 885-2573

Motorola Microcomputer Club, Steve Adler, 1301 Algonquin Rd., Schaumburg, 60196, (312) 576-3044

INDIANA
Three Rivers Users Group, Michael Milligan, 4923 Blum Drive, Fort Wayne, 46815, (219) 485-0278

Indy Color Computer Club, Mike Davis, P.O. Box 68702, Indianapolis, 46268, (317) 257-3300

Southern Indiana Computer Club, Route 1, Box 459, Mitchell, 47446

CoCo Program, Erik Merz, 310 Appletree Dr., Noblesville, 46060, (317) 842-1340

IOWA
Metro Area Color Computer Club, K.L. Knudtzon, 3324 11th Ave., Council Bluffs, 51501

Iowa City TRS-80 Users Group, Susan Chapler, R.R. 6, The Woods, Iowa City, 52240, (319) 351-5959

KANSAS
Kansas City CoCo Club, Mike Allinder, P.O. Box 11192, Kansas City, 66111, (913) 287-1904

Topeka Color Computer Club, Kevin Cronister, 2224 Hope, Topeka, 66614, (913) 272-1353

Color Computer Club of Wichita, Rex Rivers, 1205 N. Mosley, Wichita, 67214, (316) 264-9193

Walnut Valley C.C. Users Club, David Anderson, 1212 E. 4th St., Winfield, 67156, (316) 221-0040

KENTUCKY
LOCO-COCO, Mike Standefer, 3141 Doreen Way, Louisville, 40220, (502) 458-6690

LOUISIANA
Red Stick Color Computer Club, Gary Cash, 8929 Metairie Drive, Baton Rouge, 70810, (504) 293-7799

Cajun CoCo Club, Bob Hoevel, 104 Karen St., New Iberia, 70560, (318) 365-7706

MASSACHUSETTS
Greater Boston Super Color Users Group, Robert Biamonte, 6 Boulder Drive, Burlington, 01803

New England C.C. Users Group, Christopher E. Sweet, R.D. 2, Box 261, Harvard, 01451, (617) 456-8291

Massachusetts CoCo Club, Jason Rahaim, Spring St., Lunenberg, 01462, (617) 582-6514

6809'ers, Paris Nepus, 93 Grochmal Ave., #90, Springfield, 01151, (413) 732-6633

MICHIGAN
Petoskey Area CoCo Club (PAC³), Dennis Hoshield, 670 Liegl Drive, Alanson, 49706, (616) 347-0607

Greg Miller, P.O. Box 365, Haslett

Midland C.C. Club, Neil Drake, 709 Coolidge, Midland, 48640, (517) 631-2939

Michiana CoCo Club, Clay Howe, 310 S. Jefferson St., Sturgis, 49091, (616) 651-4248

MISSISSIPPI
Singing River C.C. Club, Henry Nielsen, 9001 Gray Ave., Ocean Spring, 39564

MISSOURI
North County 80 Group, Tom Vogel, 12 Ville Donna Ct., Hazelwood, 63042, (314) 739-4078

Mid-America Color Computer User's Group, Jerry Morgon, 807 Ponca Drive, Independence, 64056, (816) 796-5813

Coconuts, 1610 N. Marlan, Springfield, 65803

MONTANA
Billings C.C. Club, Jayne Kenyon, 4306 Phillip, Billings, 59101

NEW JERSEY
Intrasoft, Ravi Sakaria, 139 Candace Lane, Chatham Twp., 07928, (201) 635-8025

Bug 80 Users' Group, George R. Miller, Jr., Box 62, Glen Gardner, 08826

Loco CoCo Club, Bud Lavin, 73B Wavecrest Ave., Winfield Park, 07036

NEW MEXICO
Chaves County Color Computer Club, Lee Mitchell, 1102 Melrose Drive, Roswell, 88201, (505) 623-0789

NEW YORK
Queensboro Color Computer Club, Bob Rosen, Springfield Blvd. & 56th Ave., Bayside, 11364, (212) 631-6233

Local CoCo, Charles Martin, P.O. Box 901, Bellmore, 11710

Broome CoCo Club, Bucky Helmer, 57 Front St., Binghampton, 13905

Adirondack CoCo Club (Home Office), Bill Edwards, Box 365, Bolton Landing, 12814, (518) 644-9927

Adirondack CoCo Club (Albany Chapter), Ron Fish, Box 4214, Albany, 12204

Adirondack CoCo Club (Glens Falls Chapter), Dave and Richard Mitchell, 39 Center Street, Fort Edwards, 12828

Kings Byte CoCo Club, Morty Libowitz, 1063 East 84th St., Brooklyn, 11236, (212) 763-4233

C.C. Club of Central N.Y., Joseph Short, 248 S. Fourth Ave., Ilion, 11357, (315) 895-7730

NORTH CAROLINA
TRS-80 Users' Group of Charlotte, Jason Foodman, 240 Medearis Dr., Charlotte, 28211

Raleigh Color Computer Club, David Roper, P.O. Box 681, Garner, 27529

OHIO
CoCo Club of Youngstown, Timothy McFadden, P.O. Box 478, Canfield, 44406, (216) 788-4218

Columbus & Central Ohio Color Computer Club, D.E. Sparrow, 19 E.N. Broadway, Columbus, 43214, (614) 268-5366

Dayton CoCo Users' Group, Joseph P.Evans, 609 Applehill Dr.,W. Carrollton, 45449

Cincinnati TRS-80 Users Group, R.A. White, 44 Dow Court, Fairfield, 45014

The Cleveland CoConuts, Paul Selig, 20734, Stanford Ave., Fairview Park, 44126, 333-2920

Dayton Are Color Computer Users Group, David R. Barr, 2278 Yorkshire Pl., Kettering, 45419, (513) 293-2228

M.U.G. of CoCo—Middletown Users Group, James Carr, Middletown, 45042, (513) 424-6905

Northeastern Ohio (N.E.O.) CoCo Club, Tony Rugue, 527 Malvern Drive, Painsville, 44077, (216) 354-2736

Miami Valley CoCo Club, R. Douglas Wales, 2065 LeFeure Rd., Troy, 45373

OREGON
Willamette Valley CoCo Users, Brian James, P.O. Box 11468, Eugene, 97440, (503) 687-9286

PENNSYLVANIA
HUG-A-CoCo, George Lurie, 2012 Mill Plain Court, Harrisburg, 17110, (717) 657-2789

Penn-Jersey Color Computer Club, P.O. Box 2742, Lehigh Valley, 18001

Capital Area TRS-80 Users Group, David Morrow, 340 Lewisberry Rd., New Cumberland, 17070, BBS (717) 774-6543

Pittsburgh Color Group, William Tucker, P.O. Box 351, West Mifflin, 15122, (412) 466-3078

RHODE ISLAND
New England CoCo Nuts, P.O. Box 6604, Providence, 02940

SOUTH CAROLINA
Invitation Software Group, Tom Reed, 3562 Linbrook Dr., Columbia, 29404, (803) 786-0541

Midlands 80 Computer Club, Tommy Sanders, P.O. Box 7594, Columbia, 29202, BBS (803) 755-3414

Metropolitan Greenville CoCo Club, Ed Lowe, P.O. Box 6, Gray Court, 29645, (803) 876-3928

Hilton Head Island CoCo Club, Kevin Clark, P.O. Box 6187, Hilton Head Island, 29928, (803) 785-9630

Spartanburg County CoCo Club, Dennis Shattuck, 473 Royal Oak Drive, Spartanburg, 29302, (803) 583-3017

TENNESSEE
Chattanooga CoCo Club, Jim Perkins/Jim Cox, P.O. Box 9825, Chattanooga, 37412, (615) 870-2439

Memphis Color Computer Users Group, Billy Seaton, P.O. Box 11736, Memphis, 38111

TEXAS
CoCo User Group, David Karam, 1809 Dexter, Austin, 78704, (512) 442-6317

B/CS Color Computer Users Group, Dale Cuthbertson, 1812 Michael Lane, Bryan, 77801, (409) 822-0731

Deer Park Color Computer Club, Donald Burr, 4314 W. Grant, Deer Park, 77536, (713) 479-5313

International Color Computer Club, Inc., Robert L. Garrett, 2101 East Main Street, Henderson, 75652, (214) 657-7834

UTAH
Ogden CoCo, Kathy Rush, 4535 S. 2600 W. Roy, Ogden, 84067

VIRGINIA
D.C./N. VA. C.C. Club, Jack Darling, 43 Donovan Dr., Alexandria, 22306, (703) 780-6159

Northern Virginia C.C. Club, Logan McMinn, 6048 Belleview Dr., Apt. 1, Falls Church, 22041

Central Virginia Color Computer Club, Lane Lester, 413 Woodland Circle, Lynchburg, 24502, (804) 237-4188

WASHINGTON
Northwest Computer Club, Judy Gehman, E. 14012 Cataldo, Spokane, 99216, (509) 928-2169

WEST VIRGINIA
Kanawha Valley Personal Computer Club, P.O. Box 5354, Charleston, 25311, BBS (304) 925-3338 or 345-6502

Mtn. State CoCo Users Group, Donald G. Barber, Jr., P.O. Box 1084, Morgantown, 26507, (304) 599-4493

Mil-O-Bar C. C. Club, Jim Lemaster, Ona, 25545, (304) 743-4752

Blennerhassett CoCo Club, David Greathouse, Rt. 9, Box 119, Parkersburg, 26101

West Virginia Color Computer Club, William Mucklow, 949 Baier Street, St. Albans, 25177, (304) 727-6764

WISCONSIN
CoCo-MUG, Tom Fandre, P.O. Box 10152, Milwaukee, 53210, (414) 542-0600

Southern Wisconsin CoCo Club, David C. Buehn, P.O. Box 411, Twin Lakes, 53181

CANADA

ALBERTA
Calgary Color Computer Club, David Logan, 151 Whitelock Place N.E., Calgary, T1Y 4S7

Medley Computer and Electronics Club, P.O. Box 1267, Medley, T4A 2M0

NEW BRUNSWICK
Moncton Colour Computer Users Group, Richard Bissett, 43 Pine Glen Rd., Riverview, E1B 1V3

BRITISH COLUMBIA
North Island CoCo Club, Brian Marshall, P.O. Box 1740, Port Hardy, B.C., V0N 2P0

NEWFOUNDLAND
Avalon CoCo Club, A.R. Thompson, 10 Foran St., St. John's, A1E 4G1

NOVA SCOTIA
Halifax Dartmouth CoCo Users Group, P.O. Box 572, Dartmough, B2Y 3Y9, (902) 469-3656

ONTARIO
International Adventurer's Club, Maurice Dow, 84 Camberley Cres., Brampton, L6V 3L4

K-W C.C. Club, Peter Karwowski, 23 Hudson Crescent, Kitchener, N2B 2V7

London CoCo Nuts Computer Club, Harry K. Boyce, 180 Concord Road, London, N6G 3H8, (519) 472-7706

Meadowvale Color Computer Club, Howard Porter, P.O. Box 186, Streetsville, Mississauga, L5M 2B8

Niagara Regional CoCo Club, Harry VanDyke, 7707 Jubilee Drive, Niagra Falls, L2G 7J3

QUEBEC
Montreal TRS-80 Users Group, Jacobs P.C. Bagchus, 1176 Phillips Place, Suite 201, Montreal, H3B 3C8, (514) 861-3488

South Shore CoCo Club, Jacques Bedard, 33 Lisilre, St-Constant, P.Q., J0L 1X0

SASKATCHEWAN
Regina Color Computer Club, Georges Glass, 26 Tweedsmuir Bay, Regina, S4X 2B1, (306) 949-3942

Saskatoon Color Computer Club, Guy Tomashewski, 415-423 Pendygrasse Rd., Saskatoon, S7M 4Z2

AUSTRALIA
Blacktown City Colour Computer Users Group, Keith Gallagher, 27 Alford St., Blacktown, New South Wales, 2148

CoCo Colyteens of Colyton, Dwayne Manson, P.O. Box 506, St. Marys, New South Wales, 2760

MEXICO
Mexico City Color Computer Club, Marcelo Luft, Laja #232, 01900 Mexico D.F., Mexico City, 5-68-78-75

WEST GERMANY
First CoCo Club Hamburg, Theis Klauberg, 2000 Hamburg 62, Bultenmoor 43

new clubs

Editor:

I am extremely pleased to announce the promotion of a new CoCo club. We are a non-profit organization with more features than I think you will find in any other CoCo club. Our club library can boast the ownership of over 300 CoCo programs and we are growing daily due to new member submissions as well as our weekly BBS hunt for new downloadable programs.

Anyone wishing to find out what owning a CoCo is all about can send a self-addressed stamped envelope to: Scarborough CoCo Club, 936 S. Wallace, 71822

Ashdown, AR

Editor:

It is with great plesure that I announce the birth of southeastern Arizona's TRS-80 Color Computer Users Group. At present, we claim 12 members. All members have upgraded memory to 64K and our systems range from cassette based to disk drive, and include both the original Color Computer and the newer CoCo II. We boast several talented programmers from our ranks and have plans to submit their work to future issues of THE RAINBOW.

We welcome correspondence from other already existing CoCo groups as well as interested inquiries from local CoCo owners. Write to: Huachuca Hot Pokers, 228B Jeffords St., 85613. Call (602) 458-0292 or 458-8338.

Tim Tulp
Ft. Huachuca, AZ

Editor:

We are organizing a TRS-80 Users Group (including CoCos) in San Luis Obispo. We will meet the third Tuesday of each month in the Radio Shack Store at 209 Madonna Road. Call Wes Porter at (805) 481-2387, or write to: 2840 Halcyon Road, 93420 for further details.

Edward Bitter
Arroyo Grande, CA

Editor:

Is there any interest in forming a Bakersfield area CoCo Users Group? Meet others with common interests and exchange ideas on software, debugging, machine language, assembly language, hardware mods, programming, hints, group purchasing, amateur radio, problem solving, etc.

For information, call or write: 2521 Bishop, Apt. A, 93306, (805) 872-8618.

Larry Sheridan
Bakersfield, CA

Editor:

I would like to start a local Los Angeles Color Computer Club for us local central Los Angeles Color Computer buffs. All the present clubs are not in the city of Los Angeles. Call (213) 464-6809.

Franz Grey
Hollywood, CA

Editor:

There is a users group in the New Haven area that has started up recently. Anyone interested in coming to the meeting, please call or write to: 544 Orange Center Road, 06477, (203) 799-2631.

Patrick Palmer
Orange, CT

Editor:

The CoCo Chips has been in existence for over a year. We have a large club library and meet once a month. The meetings are held at the Montessori School in Beleair on the first Friday of each month at 8 p.m. For further information, please contact Linda Signor at (813) 581-7779 or write to The CoCo Chips Color Computer Club, 6 Belle Meade Circle, Largo, FL 33540.

Robert Fallona
Pinellas Park, Fl

Editor:

We would like to inform you that there is a CoCo Users Group in Elgin.

We of the Illinois Color Computer Club have enjoyed your magazine articles and programs enormously. A number of us attended the RAINBOWfest and thought it was even better than the first.

At the present time we have about 22 members and meet the third Wednesday of the month at Aunt Mary's Restaurant just north of the I-90 expressway on Route 25 at 7:30 p.m.

We hope to grow in the knowledge and enjoyment of being a CoCo user as you have grown bigger and better youself. Keep up the fantastic work. Write to: 1117 Hunter Drive, 60120.

Ray Krack
Elgin, IL

Editor:

Evansville area CoCo and MC-10 users are invited to help form the Evansville CoCo Users Group. If you are interested, call me at 424-0099, or Brad Callahan at 476-9453. Brad has a BBS on this number on weekends and you can leave us a message there.

Dave Jenkins
Evansville, IN

Editor:

The Toccoa Micro-computer Society is meeting on a new schedule. The meetings are every second Tuesday evening of each month at 7 p.m. at the Toccoa-Stephens County Public Library.

Thank you for this service. I realize that it costs THE RAINBOW a considerable amount to publish a Clubs section. It is very nice of you to do this for your readers. I guess that is just one of the many reasons that THE RAINBOW is miles ahead of the other CoCo magazines.

Steve Shields
Eastonollee, GA

Editor:

The Oahu Color Computer Club meets at the Liliha Public Library in Honolulu the third Thursday of every month. All interested persons are invited to call or write: 47-164 C. Hui Akepa Place, 96744, or (808) 239-6534.

Steve Galeski
Kaneohe, HI

Editor:

I would like to inform your readers of the formation of the Lexington Color Computer Users Group. We meet the second Monday of each month. Anyone wishing more information should contact me at 224 Fairdale Drive, 40505 or call (606) 255-6007 after 6 p.m.

Ed Jelf
Lexington, KY

Editor:

I would like to know if anyone in the Ferriday-Jonesville area would be interested in starting a Color Computer Club. If so, please write me at Frogmore Pltn., 71335.

George Tanner
Frogmore, LA

Editor:

I'd like to announce the founding of the Framingham Color Computer Club. We meet at Nat Walsh Music Studio on the second and fourth Thursdays of every month. There is also a BBS run on a CoCo by the club that may be reached at (617) 872-5170.

Mitch Cohen
Framingham, MA

Editor:

I would like to inform your readers that the New England Color Computer Users Group has a new address. Our new address is: N.E.C.C.U.G., R.D. 2, Box 261, 01451. We hold meetings about once every three to four weeks and we have a newsletter that comes out every month. If you would like information on the group, please contact us.

Chris Sweet
Harvard, MA

Editor:

I would like to form a Color Computer Club in my area. I own a CoCo and I think it would be great to start a club. If you are interested, please contact me at 6321 Bosker Lane, 48444.

John Sormly
Imlay City, MI

Editor:

We are currently forming a users group in the Jefferson City area and our numbers have risen to 12 interested individuals. For more information, write to: 900 Rock Hill Road, 65101.

Wayne Johnson
Jefferson City, MO

Editor:

I have had my CoCo for about three years and I am trying to start a CoCo club in the St. Louis area.

Anyone interested call or write (314) 327-6402, 856 Hancock Road, 63385.

Any age welcome!

Kel Kerr
Wentzville, MO

Editor:

The CoCo Users Group - Monmonth/Ocean County, meets on the second Friday of every month, 7:30 p.m. at 1401 Highway 71, 07719. For more information call (201) 280-2727 or 774-7426.

John Stewart
Belmar, NY

Editor:

We are proud to announce the birth of Intrasoft, an organization for Color Computerists. Members receive: a monthly newsletter, access to the Intrasoft library (software rentals through the mail), a list of members in their area, and vast amounts of information relating to the Color Computer.

For more information, call (201) 635-8025 after 3:30 p.m. on weekdays and ask for an Intrasoft representative. The first publication of our newsletter in Dec. 1, 1984. Library opens Dec. 1, 1984.

Ravi Sakaria
Chatham, NJ

Editor:

I live in Buffalo and am finding it quite difficult to find a Color Computer Club or a modem-equipped computer buddy nearby without paying a fortune in bills.

Does anyone in the Buffalo area know of any formed users group by mail or modem? There must be someone! I'd appreciate someone calling me at (716) 822-0175, or write to 33 Fields Ave., 14210.

Dan D'Alessandro
Buffalo, NY

Editor:

A few friends and I have been considering forming a Color Computer Club in Morehead City which would serve all of eastern North Carolina. The cost would be about $4 a year and would cover such expenses as newsletters, a software library containing programs written by our members and possibly a bulletin board system containing such software. If there is anyone who might be interested in such a club, please write to me at P.O. Box 943, 28532 or call me at (919) 247-3037.

Mike Fahy
Havelock, NC

Editor:

As owners of Radio Shack's TRS-80 Color Computers, we are interested in contacting other Color Computer owners. We are lo-

cated in the Asheboro-Ramseur area. Contact either John Lutsie, Rt. 9, Box 312, 27203, phone 625-5477 anytime; or phone me at 824-2612 after 3 p.m.

Helen Reynolds
Ramseur, NC

Editor:

I'd like to start a CoCo users group in the Grove City-Columbus area. Anyone interested can call me at (614) 875-9786 or write me at 2480 Northbranch Road, 43123.

Mike Gould, Jr.
Grove City, OH

Editor:

We are happy to announce the existence of the CoCo Nuts Computer Club of Portland. We invite anyone in the Portland metropolitan area to attend our meetings. We meet the first Saturday of every month at the Far West Federal Building at 45th and Woodstock. We have a very large program library on both tape and disk available to all members. We would like to exchange newsletters with any interested CoCo Clubs as we are very proud of our newsletter and very interested in finding out what other clubs are doing. Please address all inquiries and newsletters to P.O. Box 6507, 97228. Phone (503) 252-5681.

Rick Roth
Portland, OR

Editor:

We are pleased to announce the formation of the Salt City CoCo Club of Utah in Salt Lake City for users of the Color Computer.

Our group is promoting interest in the Color Computer in the Salt Lake area. Some of the areas of interest in our club are hardware experimenting, machine language programming, educating others and showing off our Color Computers to the community.

We currently publish a monthly newsletter. Articles and programs from members are included in each issue.

We meet on the even Thursdays of each month at 7 p.m. at Utah Transit Authority, 3600 S. 700 W. Anyone interested can call me at (801) 487-6032 or write to 720 E. Browning Avenue, 84105.

Dennis Mott
Salt Lake City, UT

Editor:

The Color Computer: Montgomery County Users Group (CoCo:MUG) has been formed and meets monthly at the Christiansburg Library. Our members represent a wide range of CoCo users, from educational to business applications. One member even ran a BBS, (which we are trying to convince him to restart).

Any interested parties may contact me at P.O. Box 774, 24073.

Ron Rogers
Christiansburg, VA

Editor:

Please let it be known to your readers that I am trying to start a computer club for TRS-80s. Anyone interested can contact me by writing RT. 2, Box 76-C, 25801.

Greg Reed
Beckley, WV

Editor:

I would like to announce the forming of a TRS-80 club in the New London-Hortonville area. This is a group especially for those interested in getting games and other things children would like. We will not discuss word processing and business uses, but will see how many have printers and decide to discuss that or not. We would like children from the ages 11-14, but please call for exceptions. For information write RR 4, Box 156, 54961 or call (414) 982-2565.

I would also like to have contact from other CoCo Clubs around the country, or just CoCo fans not in a club.

Paul Steinberg
New London, WI

Editor:

Over the past two years a club in Milwaukee, called CoCo-MUG, has been in full swing. For information write: CoCo-MUG, P.O. Box 10152, 53210; or call Dan at (414) 258-9448 or Tom Fandre (414) 542-0600.

Dan Neuman
Wauwatosa, WI

Editor:

A new users group is being formed, the ESSA Color Computer Club in Simcoe County, Ontario. I urge all owners or potential owners of the CoCo in the Barrie, Angus-Borden and surrounding area to come out to the club meetings.

Activities include instruction in BASIC and Assembler programming, preparing of a newsletter, general and specific discussions and any other related activities. For further information call me at (705) 424-1354 or write to P.O. Box 1001, L0M 1B0.

Eldon Doucet
Angus, Ontario

Editor:

There are a lot of Canadians up here with CoCos, and we all think that your magazine is the best!

We'd like to hear from anyone in the Bruce County area who would like to form a CoCo club or just like to share information about our favorite computer. Anyone interested can write to P.O. Box 341, N0G 2V0.

John and Mimi Wong
Walkerton, Ontario

Editor:

I am interested in joining or forming a Color Computer Club in Puerto Rico. If you have information about a club or would like to form one, please contact me at Paradis B-24, Caguas, P.R. 00625 or call (809) 743-2684.

Radames Aguayo
Caguas, P.R.

Editor:

We are proud to announce the existence of the Dutch Color Computer Club. We have given it the name COCONUT. There are almost 200 members, some of them having a Dragon computer, which we also fully support. Our library now contains approximately 2,000 programs, most of them coming from the USA.

Send correspondence to: COCONUT, c/o Peter Senff, Waterrad 104, 1613 CR Grootebroek, The Netherlands.

Peter Senff
The Netherlands

Editor:

I would like to announce that I am activating a Philatelic Users Group for stamp collectors. I have worked it out so that the membership will more than pay for itself over the course of the year. For those that are interested, write to 21531 Meyers Road, 95320.

John Root
Escalon, CA

Editor:

I would like to inform the readers about the South Bay Color Computer Club. We meet the third Friday of every month at Mercury Savings in Torrance. For more information, call (213) 376-9755 or 538-8394.

Robert Cakebread
Redondo Beach, CA

Editor:

Please inform your readers of the Dade Color Users Group of Miami. We are the largest and oldest Color Computer group in Dade County. Meetings are held at 7:30 p.m. on the third Tuesday of each month at the Miami Shores Community Center, 96th Street and Northeast 7th Avenue.

Further information may be obtained by leaving EMail on CompuServe to PPN # 70365, 463 or by calling the club BBS, 24 hours a day, (305) 751-6809.

Yearly dues are $12/individual, and $18/family.

John Lovell
Miami, FL

Editor:

The Atlanta Area Color Users Group has been in existence for three years and we now meet 10 months a year at Nash Middle School in Smyrna. Meetings begin at 7 p.m. and there are no dues at present. For more information, call Gary at 949-0369, Lee at 373-3216, or CBBS 378-4410.
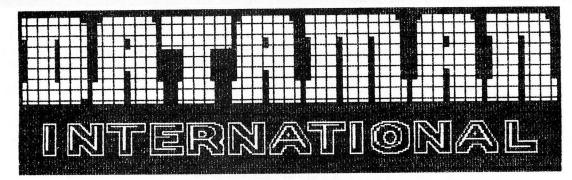
Gary McConville
Douglasville, GA

Editor:

Our computer club name is Le Club d'Ordinateur Couleur de Quebec, Inc. (C.O.C.Q.).

The meetings will be every Wednesday at 7 p.m. at C.O.C.Q., Inc., Loisirs St-Mathieu, 7110, 8e Avenue, H2A 3C4.

Jean-Marie Beaujean
Montreal, Quebec

Why Not Try a 128K Upgrade Kit or VDOS?

By Dan Downard
Rainbow Technical Editor

● *Is it possible, and if so, how big a project would it be to build a plug-compatible RAM disk for the Color Computer, i.e., a RAM disk that would plug into the second connector of the standard two drive cable (for all of us who just have one drive)? Could the standard Radio Shack controller handle a RAM disk of 256K or even 512K?*

I have several software packages that will allow me to 'invert' the foreground/ background on the amber monitor that I use with my Color Computer. The screen looks good with the black background and bright amber characters, but would look even better if the border was also black. Is there a way to make the entire screen (border and inner square) black and the characters amber? I have been told that since the software in question uses the Hi-Res screen (51 by 24) I'm stuck with things the way they are.

Dean C. Aukes
Carrollton, TX

Dean, have you noticed the 128K upgrade kits recently advertised in the THE RAINBOW? One of these will solve your problem without resorting to an actual RAM disk. Some software authors are approaching these upgrade kits as quasi-RAM disks. Another solution is to buy *VDOS* by Dr. Preble's Programs. This program performs exactly

(Dan Downard is an electrical engineer and has been involved in electronics for 24 years through ham radio [K4KWT]. His interest in computers began about five years ago and he has built several 68XX systems.)

the function you are looking for.

As far as answering your question, we know of no practical way to use the disk controller for a RAM disk. The cable and connector are designed for a serial stream of data from a floppy disk. The data from a floppy is frequency modulated and it would seem far-out to try to emulate this signal. The several RAM disk packages for other computers we have seen replace the controller as well as the drive.

Now, for your problem with the background color of your display. It is possible to have a black screen with amber characters. The software you have just doesn't offer that option. One quick solution is to purchase the *Word-Pak* accessory from PBJ. Not only will you get a black screen with amber characters, but 80 columns to boot.

● *I have noticed that there are several expansion devices (such as voice synthesizers and Radio Shack's X-pad) that either cost extra or are not available for the CoCo 2. Is this because the CoCo 2 does not supply 12 volts to the expansion port? If this is so, why not just connect a 12-volt power supply, such as Radio Shack #273-1652 to the correct pins on the connector inside the CoCo? I am not sure of which pins to connect this to, or even if this would work at all. There are probably many CoCo 2 owners that do not like spending the extra $20 for a CoCo 2 compatible voice synthesizer or the $180 for a Multi-Pak interface. With the 12 volts, one could just get a Y-cable to connect two items to the expansion port instead of laying down big bucks for*

the Multi-Pak.

Also, could two or more Y-cables be used to get several items connected to the CoCo, such as a disk drive, the X-pad and a voice synthesizer, to be used all at the same time?

Mitch Cohen
Framingham, MA

It is very easy, and practical, to attach a 12-volt power supply to your CoCo 2, Mitch. An article on how to do this appeared in Tony DiStefano's column in the April, 1984 RAINBOW. If you already have a 12-volt power supply, connect the +12 lead to Pin 2 of the expansion connector. The ground connection should be made to Pin 33 or Pin 34.

If a regulated supply is not necessary for the device you have in mind, 12 volts is available on the positive side of C3, CR3 and CR4 in the CoCo 2. Some people have been known to run their older disk controllers requiring 12 volts by connecting this line to Pin 2 of the expansion connector.

I cannot directly answer the second part of your letter dealing with multi Y-cables without knowing what the specific devices and loading characteristics you are contemplating. Keep in mind that the expansion bus on the CoCo has unbuffered lines, e.g., direct lines to the microprocessor. Putting too much load on these lines could damage the 6809E. That's why we have the Multi-Pak interface.

● *I have just discovered your very interesting magazine in my country, and am happy to read it every month.*

I have a 32K Extended Color BASIC computer. Could you tell me how random numbers can be generated in machine language? Is there a subroutine in ROM that can be called?

J. Kostense
Rotterdam, Netherlands

Nice hearing from the Netherlands, Mr. Kostense. There is a random number generating routine in the BASIC ROM. It is located at address $BF1F. Floating point math is used and we hope to have a complete explanation of how the CoCo uses this system in an upcoming issue. The explanation is too involved for this column.

● *I have two questions involving graphics. The first is about sprite graphics. Your reply to Theodore Arrington's letter implied that to have sprite graphics you need special hardware. I have used sprites on my friend's Commodore 64 and, from what I have seen, it seems like you could do the same thing with a software feature built into BASIC. How does a sprite video display generator chip work and how do you access it?*

My second question concerns graphics Adventures. I understand how you would make one with a disk system. You would design your screens and save them as binary files. On a cassette system the screens would have to be resident in memory. You would need 6K for graphics memory, which leaves you almost enough memory for nine rooms; that leaves no room for the actual Adventure. You could use BASIC statements to draw the screens as you go along, but not only would this slow down the program, it would use up just as much memory if the screens were detailed. How can cassette-based graphics Adventures store so much data into so little memory?

The Snyders
Mansfield, CT

Sprite graphics are indeed hardware generated and not possible with the 6847 VDG inside our CoCos. You are correct in assuming that sprites can be simulated with software. In fact, this is what the Extended BASIC commands *GET* and *PUT* try to accomplish.

The advantage of having hardware sprites is that less programming is required to move objects around the screen. Since the software overhead is minimal, quicker response is possible. You have to weigh this fact with the relative speed of the microprocessor though, and with our 6809E we've managed to ignore sprites so far and still have some outstanding games.

As far as saving graphics screens for games on tape, I would recommend a copy of *Bjork Blocks* distributed by Moreton Bay Software. As this program will more than adequately demonstrate, the data on graphics screens is usually highly repetitive, and can be condensed into smaller blocks.

● *In reference to the November issue of your column, a letter from John Ruzicka of Florham Park, N.J., indicated he was experiencing many crashes, I/O Errors and directory crashes.*

The problems he was experiencing were probably related to the "Tandy Gold" used on his edge connectors on the disk controller. The tin/lead contacts oxidize, causing intermittent contact problems. The alternate directory and backups are attacking the symptom, not the problem. Our Gold Plug 80, which has been on the market for several years solving the problem on the Model I computer, and for about a year and a half on the CoCo market, will eliminate the crashes and I/O Errors. The cost is minimal, and installation is fairly simple compared to trying to reconstruct one crashed directory. The Gold Plug 80 will solve the problem, not cure the symptom.

Ed Pruitt
Keller, TX

Thanks for the information, Ed. I agree with your evaluation and would recommend gold controller connectors. At the same time, it is only fair to mention that these plugs require soldering to your disk controller edge connector.

● *I am interested in how the CoCo works. I like disassembling ROMs and using some techniques in my own assembly language programs. I am a beginner at assembly language, although I'm learning fast.*

Because BASIC has to be "translated" into machine code, is there a way to intercept the machine code produced, and use it as an assembly language program? If so, would this method speed up the program?

Tim Richter
Ulster Park, NY

Tim, I believe you may be confused by the difference between a compiler and an interpreter.

The CoCo ROMs contain an interpreter. Each line of BASIC is analyzed as it is encountered, and the commands are executed by calling the proper machine language subroutine in ROM. The only reason you can't read your program in memory exactly as you typed it is due to the memory (and time) conserving method of "tokenizing" each line. "Tokenizing" refers to the substitution of all commands with a single byte (sometimes two) equivalent.

Another method of generating BASIC programs is by using a compiler. A compiler actually translates the BASIC lines into machine language before the program is executed. This results in considerably faster execution, but normally requires more memory. There are several BASIC compilers on the market for the CoCo for you to experiment with, Tim.

● *I am writing with two questions and a tip for those who wish to take advantage of it. The tip involves some cutting and soldering so some people will not want to make the modification. Those who do will also need to modify their joysticks or construct new ones.*

It seems that when Radio Shack designed the Color Computer they decided to use a six-pin socket for the joystick ports, while the joysticks themselves use only a five-pin plug. This seems rather useless on first consideration; however, for those who have such programs as Graphicom that use the right joystick and fire button, and the left fire button, an idea springs to mind.

If you look at the way the sockets are mounted on the board, you can see that the center pin socket is connected to ground. This center mounting pin on each socket can be cut, which will allow the fire button on each port to be cross-wired to the opposite socket. Once this has been accomplished, six-pin plugs, six-conductor wire, and a soft-touch momentary switch can be picked up at other than Radio Shack (except for the switch) electronic stores, and the joysticks rewired with both fire buttons on each joystick.

Now for the questions. I have noticed several programs lately that allow for the adjustment of the step speed of disk drives. I would like to know how this is done and if it can be accomplished through BASIC commands. Possibly the Poke? Also, how is the SAM chip programmed to flip between one 32K memory bank and the other? I can flip to the all RAM mode, but am not sure about exchanging upper and lower 32K banks without physically changing the jumper inside the machine.

Thank you kindly for paying attention, and for being part of a fine publication.
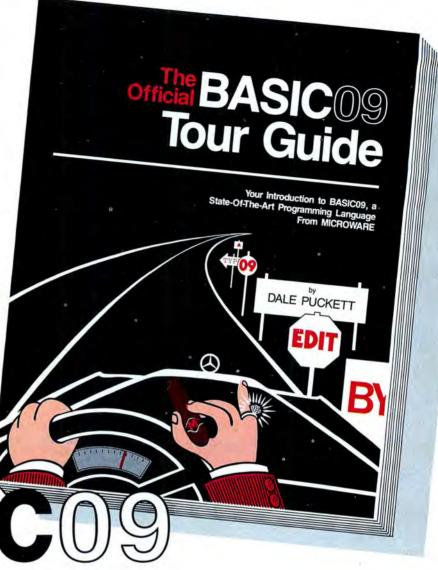
Bob Goudie
Kent, WA

Thanks for the hint, Bob. Where there's a will, there's a way.

To check the speed of your disk drives without buying OS-9, look in your August 1984 RAINBOW for an excellent program called *Disk Drive Speed Check*, by Roger Schrag. This program times the cycle produced by the index hole in the disk and converts the period encountered to RPM. Your disk drives should run at 300 RPM.

The SAM chip is programmed to page between two 32K banks of RAM by the use of the MA7 address line going to the RAM chips along with the chip select circuitry. This memory management function is all but invisible to us, as all we have to do is program the proper register to page the memory.

Your technical questions are welcomed. Please address them to: Downloads, THE RAINBOW, P.O. Box 385, Prospect, KY 40059. We reserve the right to publish only questions of general interest and to edit for space and clarity.

PERSONABLE PASCAL

Block Structure And Recursion

By Daniel Adams Eastham
Rainbow Contributing Editor

For the last couple of months we have been writing programs that contain procedures and functions. Using procedures and functions allows us to break up the overall program into smaller pieces and concentrate on solving a piece of the problem at a time. However, when we use procedures and functions there is much more than this simple dividing process that is going on.

Block Structure

The body of a program, procedure or function, is referred to as a *block*. This includes both the declaration and the execution statements. While in the past we have only shown procedures and functions contained within a program, these procedures and functions can themselves have internal procedures and functions. For example:

(Daniel Eastham holds a B.S. in computer science and has 13 years experience in systems and communications programming including work on the original Telenet packet switching network. He is the author of the DEFT Pascal Workbench *and is currently president of Deft Systems, Inc.)*

```
PROGRAM Main;

  PROCEDURE First;

    FUNCTION LocalRoutine : Real;
    BEGIN
    END;

  BEGIN (* PROCEDURE First *)
  END;

  PROCEDURE Second;

    PROCEDURE LocalRoutine;
    BEGIN
    END;

    FUNCTION SecondFunc : Real;
    BEGIN
    END;

  BEGIN (* PROCEDURE Second *)
  END;

BEGIN (* PROGRAM Main *)
END.
```

This program contains two procedures: *First* and *Second*. The procedure *First* contains a single function *Local Routine*. The procedure *Second* contains a procedure also named *Local Routine* as well as a function named *SecondFunc*. You can see where the corresponding BEGINs are for each function by the indentation level and comments beside the BEGINs.

Scoping

In this example, only the procedures *First* and *Second* are known within the main program. All procedures and functions contained within these two procedures are local to those individual procedures and are hidden from the main program.

This means that the function *Local Routine* in the procedure *First* is known only within the procedure *First*. The procedure *LocalRoutine* and the function *SecondFunc* in the procedure *Second* are both known only within the procedure *Second*.

The area (block) within which a particular symbol (label, constant, type, variable, procedure or function) is known is called its *scope*. There are basically three *scoping rules* that determine exactly where a symbol is known:

1) A symbol is known starting at the point at which it is declared.
2) It is known only in the block within

which it is declared.

3) It is known within *nested* blocks as long as there are no symbols declared within those *nested* blocks which have the same name.

Using the first rule we can see that the procedure *First* is known to the procedure *Second*. However, the procedure *Second* is *not* known to the procedure *First*. This is due to the fact that *First* is declared before *Second*.

The second rule is what allows us to use the symbol *LocalRoutine* twice: once in *First* and once in *Second*. The *LocalRoutine* in *First* becomes unknown once the end of the *First* procedure is reached. It is this second rule that we have been using all along to define local variables.

We will have to use a new example to explain the third rule:

```
PROGRAM Test;

   VAR I, J : Integer;

PROCEDURE DoSomething;

   VAR I : Real;

   FUNCTION Compute (J : Real) : Real;

   BEGIN
      I := J;
   END;

BEGIN (* DoSomething *)
   I := J;
END;

BEGIN (* Test *)
I := J;
END.
```

In this example, the symbols 'I' and 'J' are each declared more than once in the program. In the main program *Test,* 'I' and 'J' are each integers declared in the main program. In the procedure *DoSomething,* 'J' is still the same integer declared in the main program, but 'I' is now a local real number variable. In the function *Compute,* 'I' is still the real number variable delared in *DoSomething,* but 'J' is now the real parameter to the function.

You can see from this that each of the *I := J* assignment statements does something different. In the first one

(*Compute*), the local real parameter is assigned to the real variable declared in the enclosing procedure. In the second one (*DoSomething*), an integer declared within the program is converted to real and assigned to the local real variable. In the last one (*Test*), one integer is assigned to another integer.

These scoping rules are generally the same ones used in all *block structured* languages. Once you have learned them for PASCAL, you will also know them for PL/1, ALGOL, ADA and C.

Variable Allocation

In addition to only being known within the block in which it is declared, variables only *exist* when the block within which they are declared is active. Anytime a procedure or function is invoked that has some local variables or value parameters declared, memory is allocated for them at that time. When the procedure or function returns to the caller, the memory for the local variables is released. This form of memory allocation is known as *automatic* allocation since the memory for the variables is automatically allocated and released.

Automatic allocation results in two interesting properties of variables. First, the value that a variable has when its block is activated will not necessarily be the same value it had when that block was last activated. In fact, standard PASCAL requires that each variable be marked as undefined on entry to a block. This means that you cannot leave a value stored in a variable before exiting from a procedure and expect to use it the next time the procedure is entered.

Recursion

The second property results from the ability of a procedure or function to invoke itself. A procedure or function can do this by either invoking itself directly, or indirectly by invoking a second procedure, which eventually causes the first one to be invoked again. A procedure or function which does this is said to be *recursive.*

When a procedure or function is recursively activated, each activation of that procedure or function has its own set of local variables and value parameters. For example:

```
FUNCTION Factorial (Number :
   Integer) : Integer;
BEGIN
   IF Number < 0 THEN Factorial := 0
   ELSE IF Number=0 THEN Factorial
```

```
   := 1
   ELSE Factorial := Number *
   Factorial (Number - 1);
END;
```

Factorial is an integer function which returns the factorial of its parameter. In this case, *Number* is a value parameter which is allocated each time the function is activated.

For example, if the initial call is with the number 5, *Number* will be allocated and the value 5 will be stored in it. As you can see from the IF statement, this will cause *Factorial* to invoke itself with the value of 5-1 or 4. This causes *Number* to be allocated again with the value 4 being stored into it.

This process continues until there are six separate memory locations allocated for *Number* with the values 5, 4, 3, 2, 1 and 0. At this point the function does not invoke itself again, but rather returns the value 1. This, in turn, causes the multiplication of each of the other activations to complete until the final value (1*1*2*3*4*5 = 120) is computed.

Note that *reference* parameters do *not* get allocated when a procedure or function is invoked. This is because they refer directly to a variable named by the caller. Only *value* parameters and local variables are automatically allocated.

Example Program

Although the above example demonstrates how recursion works, a recursive technique is not the best method for computing factorials. Instead, an *iterative* technique (using a *FOR* loop) would normally be used. However, when dealing with computer languages or artificial intelligence, recursive techniques can be very helpful.

This month we have a direct entry calculator program. This program lets the user directly enter a general arithmetic expression which is then evaluated using precedence rules (multiplication and division before addition and subtraction), and the use of parentheses. For example, it can correctly handle the following expressions:

```
5 + 3 * 4                 (= 17)
(5 + 3) * 4               (= 32)
(5 + 3) / (4 - 1)    (= 2.6666. . .)
```

It is also able to determine if an erroneous expression was entered, and indicate exactly where in the expression the error occurred. The technique used in the program is called *recursive descent.* To use this technique we first develop a grammar for the expressions

that we are going to parse:

<expression> ::= <term>
  [<addop> <term>]
<term>::=<factor>
  [<mulop> <factor>]
<factor>::= <number> |
  (<expression>)
<addop>::= + | –
<mulop>::= * | /
<number>::= legal PASCAL
  real number

In this case an expression is one or more terms separated with addition operators (either + or -). A term is one or more factors separated with multiplying operators (either * or /). A factor is either a real number or an expression enclosed in parentheses. This last definition makes the grammar recursive. That is, the grammar is defined in terms of itself.

The program is written with a function for each level of expression. An *Expression* function adds and subtracts terms, a *Term* function multiplies and divides factors, and a *Factor* function evaluates real numbers and handles expressions within parentheses.

The name *recursive descent* comes from the fact that we descend down the grammar until we reach a place that references one of the higher elements in the grammar. At this point we recursively invoke the higher level routine that handles this element. In this case, it is the *Factor* function that invokes the *Expression* function, causing *Expression* to be recursively activated.

In addition to using recursion, this program also takes advantage of PASCAL's block structure by placing the *Factor* function inside the *Term* function, which itself is in the *Expression* function. This means that only *Expression* is known in the main program. This prevents a programmer from accidentally invoking the *Term* or *Factor* functions.

If you have any questions about block structure or recursion in PASCAL, you may call (301) 253-1300 during normal business hours for help. Next month we will go into detail about pointers and the heap in PASCAL.

If you have a disk system, these commands are needed prior to loading the program:

PCLEAR 1
FILES 0,0
CLEAR 16, 4999

The listing:

```
00 0000  (*************************************************
00 0000  *
00 0000  *   This program uses a recursive descent technique to
00 0000  *   evaluate arithmetic expressions.  After executing the
00 0000  *   program, just enter an arithmetic expression and ENTER.
00 0000  *   It will display the answer or the word ERROR and indicate
00 0000  *   where the error is.  To exit, just hit ENTER at the prompt.
00 0000  *
00 0000  *************************************************)
00 0000
00 0000  PROGRAM Calculator (Input, Output);
00 0000
00 0000  VAR CharCount : Integer;
00 0000      Value    : Real;
00 0000      ThisChar : Char;
00 0000
00 0000  (*************************************************
00 0000  *
00 0000  *   Routine to Parse an Expression
00 0000  *
00 0000  *************************************************)
00 0000
00 0000  FUNCTION Expression (VAR Result : Real) : Boolean;
10 0003
10 0003  VAR TermAmount : Real;
10 0003      Operator   : (OpAdd, OpSubtract);
10 0003
```

```
10 0003  (*************************************************
10 0003  *
10 0003  *   Local Routine within Expression to Parse a Term
10 0003  *
10 0003  *************************************************)
10 0003
10 0003  FUNCTION Term (VAR Result : Real) : Boolean;
20 0006
20 0006  VAR FactorAmount : Real;
20 0006      Operator    : (OpMultiply, OpDivide);
20 0006
20 0006  (*************************************************
20 0006  *
20 0006  *   Local Routine within Term to Parse a Factor
20 0006  *
20 0006  *************************************************)
20 0006
20 0006  FUNCTION Factor (VAR Result : Real) : Boolean;
30 0009
30 0009  VAR MinusFlag  : Boolean;
30 0009      Number     : String;
30 0009  BEGIN
31 0010    Factor := False;
31 0014    MinusFlag := False;
31 0018
31 0018    REPEAT
31 0018      READ (ThisChar);
31 0032      CharCount := CharCount + 1;
```

HARDWARE & PROGRAMS

MONITORS

BMC MEDIUM-RES COLOR

13" BMC w/ sound $303.95
14" USI w/ sound 324.95
12" Taxon Composite & RGB. . . 335.95

COMREX HI-RES MONOCHROME

12" Amber or Green. 140.95
9" Amber or Green. 125.95

Sorry, no C.O.D. on monitors.

COMPOSITE MONITOR INTERFACES

Double Driver. 24.95
Video Plus. 24.95
   Both work great with color
   or monochrome on CoCo I.

Coco Double Driver. 28.95
Video Plus II M 26.95
Video Plus II C. 39.95
   For CoCo II Only

JARB DISK DOUBLER

Why spend twice as much as you need to for double sided diskettes? With our doubler, you can make your own and pay for it with the first box you double. A must for disk drive users.
5¼" size only 12.95

BASF DATA CASSETTES

	C-05	C-10
1-10	.60 ea.	.65 ea.
11-20	.55 ea.	.60 ea.

Soft Poly Cases . . . . Ea. $.20
Hard Shelled Cases . . . . Ea. $.22
Cassette Labels (12). . . . . Sh. $.36
Cassette Labels Tractor (1000) $21.95

MEMORY UPGRADE KITS

16K RAM CHIPS . . . . 1.50 ea. .
5V, CoCo II 16K . . . . 1.95 ea.

'64K RAM CHIPS

Eight 200 NS Factory Prime 64K RAM Chips. Allows you to upgrade 'all' board easily. No soldering needed. $52.50

'16K/32K

Eight 200 NS Factory Prime Chips with Piggy Backed Sockets, Sam Socket, Bus Wire. Comprehensive Instructions. Recommended for "D" or earlier, but may be used on "E". Only 9 simple solder connections to kit. None to computer. $25.95
NOT FOR CoCo 2

NEW

©JARB SOFTWARE 1984

THE GUNFIGHTER

BY Terry A. Steen

An excellent hi-res, arcade quality game program for two players. Joysticks and 32K are required in this all machine language program.
Cassette . $19.95   Disk/Amdisk . $24.95

JUNGLE TREK

Lost in a jungle with wild animals lurking; your only survival is to find a safe compound before you are lunch for lions; high resolution; multi-color.
16K EXT $14.95

BIORHYTHM/PSYCHIC APT.

1) Prints biorhythm charts of nearly unlimited length; attractively formatted for use on most printers.   16K

2) Your psychic ability is determined through questions evaluating your psychic experiences
16K Ext . . . . Both for $15.95

PROGRAMS FOR THE SPECTRUM PROJECTS VOICE-PAK OR SPEECH SYSTEMS VOICE

TALKING FINAL COUNTDOWN

(by Bill Cook)

For 32K EXT $19.95
Standard cassette
FINAL COUNTDOWN $14.95

TALKING SPELL-A-TRON

The program allows the user to build a dictionary of words. During testing, the words are spoken. If an incorrect response is given, the word is spoken again and spelled. Tape (32K EXT) $22.95

TALKING SCORE E-Z

A yahtzee type program. Up to six players can compete. All scoring and record keeping is done by the computer. Tape (32K EXT) $19.95
Standard SCORE E-Z $15.95

TALKING COLOR MATH

The perfect educational game to aid the student in learning addition, subtraction, multiplication and division. Allows one to specify difficulty level.
Tape (32K EXT) $22.95

TALKING SHIP HUNT

by Cobra Software

Play Battleship against your computer. 32K w/ joystick needed. Graphics and sound. Can be played without voice.
Cassette $10.95

SCHEMATIC DRAFTING PROCESSOR (disk) can draw large scale schematics in hi-res (has six overlapping screens) and then print them out to any of several popular printers, fast!! A must for serious hardware computerist.
Now only. $49.95

CoCo Chips

Sam, Pia, CPU, Ext, Basic

We carry products from many manufacturers. If you don't see it, ask.

```
31 0043    UNTIL ThisChar <> ' ';
31 0050
31 0050    IF (ThisChar = '-') OR (ThisChar = '+') THEN BEGIN
32 007E      MinusFlag := ThisChar = '-';
32 0091      REPEAT
32 0091        READ (ThisChar);
32 00AB        CharCount := CharCount + 1;
32 00BC      UNTIL ThisChar <> ' ';
32 00C9      END;
31 00C9
31 00C9    CASE ThisChar OF
31 00D1      '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
31 0122      '.' : BEGIN
32 012A            Number := '';
32 0137            REPEAT
32 0137              Number := Number + ThisChar;
32 0154              READ (ThisChar);
32 016E              CharCount := CharCount + 1;
32 017F            UNTIL NOT (ThisChar IN ['0'..'9', '.', 'E']);
32 01B2            Result := ENCODEREAL (Number);
32 01D0            END;
31 01D3      '(' : BEGIN
32 01DB            IF NOT Expression (Result) THEN EXIT;
32 0202            IF ThisChar <> ')' THEN EXIT;
32 021B            READ (ThisChar);
32 0235            CharCount := CharCount + 1;
32 0246            END
31 0246      ELSE EXIT
31 024D      END;
31 024D
31 024D    IF MinusFlag THEN Result := -Result;
31 0267
31 0267    WHILE ThisChar = ' ' DO BEGIN
32 027C      READ (ThisChar);
32 0290      CharCount := CharCount + 1;
32 02A1      END;
31 02A3
31 02A3    Factor := True;
31 02A8    END;
```

```
20 02AC    (************************************************************
20 02AC    *
20 02AC    *   Main Entry for Term Routine
20 02AC    *
20 02AC    ************************************************************)
20 02AC
20 02AC    BEGIN
21 02B2      Term := False;
21 02B6      IF NOT Factor (Result) THEN EXIT;
21 02D7
21 02D7      WHILE ThisChar <> CHR (13) DO BEGIN
22 02EB        CASE ThisChar OF
22 02FD          '*' : Operator := OpMultiply;
22 02FD          '/' : Operator := OpDivide
22 0305          ELSE BEGIN
23 030D                Term := True;
23 0312                EXIT;
23 0316                END
22 0316          END;
22 0316
22 0316        IF NOT Factor (FactorAmount) THEN EXIT;
22 0336
22 0336        CASE    Operator OF
22 0338          OpMultiply : Result := Result * FactorAmount;
22 035E          OpDivide   : Result := Result / FactorAmount
22 0373          END;
22 0380        END;
21 0383
21 0383      Term := True;
21 0388    END;

10 038C    (************************************************************
10 038C    *
10 038C    *   Main Entry for Expression Routine
10 038C    *
10 038C    ************************************************************)
10 038C
10 038C    BEGIN
11 0392      Expression := False;
11 0396      IF NOT Term (Result) THEN EXIT;
11 0397      WHILE ThisChar <> CHR (13) DO BEGIN
12 03C9        CASE ThisChar OF
12 03CB          '+' : Operator := OpAdd;
12 03DB          '-' : Operator := OpSubtract
12 03E3          ELSE BEGIN
13 03EB                Expression := True;
13 03F0                EXIT;
13 03F4                END
12 03F4          END;
12 03F4
12 03F4        IF NOT Term (TermAmount) THEN EXIT;
12 0414        CASE    Operator OF
12 0416          OpAdd      : Result := Result + TermAmount;
12 043C          OpSubtract : Result := Result - TermAmount
12 0451          END;
12 045E        END;
11 0461
11 0461      Expression := True;
11 0466    END;

00 046A    (************************************************************
00 046A    *
00 046A    *   Main Entry for Calculator Program
00 046A    *
00 046A    ************************************************************)
00 046A
00 046A    BEGIN
01 0470      Page;
01 047C      WRITELN;
01 0484      WRITELN ('CALCULATOR PROGRAM');
01 04A8
01 04A8      REPEAT
01 04A8        WRITE ('> ');
01 04BB        CharCount := 0;
01 04BF        IF Expression (Value) AND (ThisChar = CHR (13)) THEN
01 04EC          WRITELN ('=', Value:14:4)
01 0512        ELSE IF (CharCount > 1) OR (ThisChar <> CHR (13)) THEN BEGIN
02 0546          WRITELN (' ':CharCount, ' ^ ERROR');
02 056C          IF ThisChar <> CHR (13) THEN READLN;
02 0584          END;
01 0584      UNTIL (CharCount = 1) AND (ThisChar = CHR (13));
01 05B2    END.
```

hogg_wash

More 128K

By Frank Hogg
Rainbow Contributing Editor

Last month, I promised I would have a program to use the 128K with FLEX. I am happy to report I have done just that. There is always a chance something will interfere with my plans, and I am glad it didn't, so on to the program.

When I first thought of doing this, I planned to use a command that worked like so: BANK alone would give you the current bank we are in. BANK 1 would switch to bank 1, BANK 1, 100 would switch to bank 1 and jump to address Hex 100. This turned out to be clumsy. First off, you never know what bank you are in unless you type BANK, and this is a pain. Second, the length of the command line is too long for heavy use. Here's an alternative. There are three commands, 'a', 'b' and 'c', all in the one program below. All of them do the same thing — they switch to either bank 0, 1 or 2. I used letters instead of numbers because they look better as the first character of the prompt. When you switch to bank 'b', for instance, the first letter of the '+++' prompt becomes a 'b', like so: 'b++'. This way, you always know what bank you

are currently in. If you put a Hex number after the command, you will jump to that address. This works just like the FLEX JUMP command.

The entire program fits inside an area of memory that is empty in CC FLEX 5.0:4. The area at $FE00 to $FEFF is not used and is therefore usable for this purpose.

When first issued, BANK adds itself to FLEX's user command table and a small program initializes everything. You will see the prompt change from '+++' to 'a++' to show that we are in bank 'a'. Try typing 'b' or 'c'. Instantly the prompt changes and so does the bank.

I am presuming you have an installed and tested 128K board in your system. This program does not check for that; it just assumes it and will look the same on a system with only 64K. Let's try it out.

```
a++B
b++GET 0.ED.CMD
b++C
c++GET 0.ASM.CMD
c++A
a++
```

We now have ED in bank 'b' (1) and ASM in bank 'c' (2). This allows us to edit a program using bank 'b', assemble it using bank 'c', and then test it using bank 'a'! All without accessing the disk,

except to read the file we are working with. Here's how it looks in practice.

```
a++B 0 test
b++C 3 test
c++A
a++test
```

'B 0 test' switched to bank 'b', jumped to Hex 0 where the cold start address of ED is and edited the file test. 'C 3 test' switched to bank 'c', jumped to Hex 3 where the cold start address of ASM is and assembled the program test. Then we switched to bank 'a' and ran the program to try it out. This speeds up program development like you wouldn't believe!

The pitfalls: It is important to have a program in memory before jumping to it. I speak from experience. The results can be amusing if you try it without a program in memory. One time I did this; the printer started to issue formfeeds one after another while the screen went bananas. The lesson is to make sure to have a program in the bank before you jump into it. In other words, don't jump off the cliff without someone to catch you.

Advanced Uses

How would you like to have several programs running in separate banks and switching from one to the other? Well, I took the challenge by trying to

have two XBASIC programs that called each other. I had to add some things to the program to save the current stack. I think you could expand on what I have done, but this does work.

First, the problems; FLEX has 48K of user memory and these banks are only 32K. That means the upper 16K of the user memory cannot be switched, only the lower 32K. Many programs use whatever user memory is available, so this could be a problem. The answer is that these same programs use FLEX's memory end pointer (MEMEND) to restrict themselves to the area available. All we have to do is set MEMEND to $7FFF, which indicates to these programs that only the lower 32K is available to them. Use the MEMEND command to set MEMEND to $7FFF, and we are all set.

```
a++MEMEND 7FFF
7FFF
a++B
b++XBASIC
```

READY
10 PRINT "I AM IN BANK B"
20 EXEC, "C:XBASIC"
30 PRINT "I AM BACK IN BANK B"
RUN
I AM IN BANK B
READY
10 PRINT "I AM IN BANK C"
20 EXEC, "B"
30 PRINT "I AM BACK IN BANK C"
40 GOTO 20
RUN
I AM BACK IN BANK B
READY
20 EXEC, "C"
40 GOTO 20
LIST
10 PRINT "I AM IN BANK B"
20 EXEC, "C"
30 PRINT "I AM BACK IN BANK B"
40 GOTO 20
RUN
I AM IN BANK B

I AM BACK IN BANK C
I AM BACK IN BANK B
I AM BACK IN BANK C
I AM BACK IN BANK B
I AM BACK IN BANK C
I AM BACK IN BANK B
I AM BACK IN BANK C
I AM BACK IN BANK B
I AM BACK IN BANK C
I AM BACK IN BANK B
I AM BACK IN BANK C
I AM BACK IN BANK B
I AM BACK IN BANK C
I AM BACK IN BANK B

Stop the thing by holding down the BREAK key and hitting ENTER. You could probably think of something more useful than this, but you get the idea. It would be possible to use all three banks this way, and you could even switch MEMEND up and down in one of the banks. The possibilities are interesting. At worst, you have several new ways to crash your system and, at best, you can make a powerful computer even more useful than before.

Listing 1:

```
0                nam BANK.CMD
1                    * Bank switch and jump command for DSL 128K on CoCo
2
3       CD42    GETHEX  equ  $cd42
4       CD03    WARMS   equ  $cd03
5       CC12    UTABLE  equ  $cc12
6       CC16    ESCREG  equ  $cc16
7       CC4E    plus    equ  $cc4e    First '+' in '+++'
8       FE00    open    equ  $fe00
9       FFAC    latchc  equ  $ffac
10      FFAD    latchd  equ  $ffad
11      FFAE    latche  equ  $ffae
12      FFAF    latchf  equ  $ffaf
13
14 FE00                 org  $fe00
15 FE00 00      bank    fcb  0          Current bank we are in
16 FE01 0000    esctab  fdb  0          Bank 0 ESC return address
17 FE03 0000            fdb  0          Bank 1 etc
18 FE05 0000            fdb  0          Bank 2 etc
19                  * Stack pointer holding area
20 FE07 0000    ststab  fdb  0        0
21 FE09 0000            fdb  0        1
22 FE0B 0000            fdb  0        2
23                  * User command table extension
24 FE0D 41      table   fcc  'A'      0
25 FE0E 00              fcb  0
26 FE0F FE1E            fdb  a
27 FE11 42              fcc  'B'      1
28 FE12 00              fcb  0
29 FE13 FE3E            fdb  b
30 FE15 43              fcc  'C'      2
31 FE16 00              fcb  0
32 FE17 FE63            fdb  c
33 FE19 50              fcc  'P'      Existing P command
34 FE1A 00              fcb  0
35 FE1B E3B8            fdb  $e3b8
36 FE1D 00              fcb  0          Ends table
37
38                  * Bank 0 (a command)
39
40 FE1E 34 10   a       pshs x          dummy
41 FE20 34 7F           pshs u,y,x,dp,b,a,cc save it all
42 FE22 8D 75 FE99      bsr  abc
43 FE24 10AE 84         ldy  0,x        get bank 0's return address
44 FE27 10BF CC16       sty  ESCREG     restore it
45 FE2B 8E FE07         ldx  #ststab    point to stack table
46 FE2E 10EE 84         lds  0,x        reset stack pointer
47 FE31 7F FE00         clr  bank       set bank to a
48                  * Switch to 0
49 FE34 7F FFAE         clr  latche
50 FE37 86 61           lda  #'a        set prompt
51 FE39 B7 CC4E         sta  plus
52 FE3C 20 48 FE86      bra  jump       go
53
54                  * Bank 1 (b command)
55
56 FE3E 34 10   b       pshs x          dummy
57 FE40 34 7F           pshs u,y,x,dp,b,a,cc
58 FE42 8D 55 FE99      bsr  abc
59 FE44 10AE 02         ldy  2,x
60 FE47 10BF CC16       sty  escrr
61 FE4B 8E FE07         ldx  #ststab    point to stack table
62 FE4E 10EE 02         lds  2,x        reset stack pointer
63 FE51 86 01           lda  #1
64 FE53 B7 FE00         sta  bank       set to bank b
65                  * switch to 1
66 FE56 7F FFAC         clr  latchc
67 FE59 7F FFAF         clr  latchf
68 FE5C 86 62           lda  #'b
69 FE5E B7 CC4E         sta  plus
70 FE61 20 23 FE86      bra  jump
71
72                  * bank 2 (c command)
73
74 FE63 34 10   c       pshs x          dummy
75 FE65 34 7F           pshs u,y,x,dp,b,a,cc
76 FE67 8D 30 FE99      bsr  abc
```

```
77  FE69 10AE 04          ldy     4,x
78  FE6C 10BF CC16        sty     escrr
79  FE70 8E   FE07        ldx     #ststab  point to stack table
80  FE73 10EE 04          lds     4,x      reset stack pointer
81  FE76 86   02          lda     #2
82  FE78 B7   FE00        sta     bank     set to bank c
83                    * switch to 2
84  FE7B 7F   FFAD        clr     latchd
85  FE7E 7F   FFAF        clr     latchf
86  FE81 86   63          lda     #'c
87  FE83 B7   CC4E        sta     plus
88                    * Jump to address on command line if there
89  FE86 BD   CD42  jump  jsr     GETHEX   Get address if there
90  FE89 25   07   FE92   blo     exit     no address
91  FE8B 5D              tstb
92  FE8C 27   04   FE92   beq     exit
93  FE8E AF   6A          stx     $A,s     Where dummy was put
94  FE90 35   FF          puls    cc,a,b,dp,x,y,u,pc pull and goto address
95  FE92 8E   CD03  exit  ldx     ##cd03
96  FE95 AF   6A          stx     $A,s     Also where dummy was put
97  FE97 35   FF          puls    cc,a,b,dp,x,y,u,pc (jmp WARMS)
98
99
100 FE99 8E   FE07  abc   ldx     #ststab  Point to table
101 FE9C B6   FE00        lda     bank     get old bank
102 FE9F 48              asla            mul x 2
103 FEA0 10EF 86          sts     a,x      store stack pointer for this bank
104 FEA3 8E   FE01        ldx     #esctab  Point to ESC table
105 FEA6 10BE CC16        ldy     ESCREG   get return address
106 FEAA 10AF 86          sty     a,x      save it for return to this bank
107 FEAD 39              rts
108
109       00AE     size  equ     *-$fe00  Show size of main program
110                   * Overlay user command table in FLEX
111 CC12                  org     UTABLE
112 CC12 FE0D            fdb     table
113
114                   * Initialize routine (called only once)
115 C100                  org     $c100
116 C100 7F   FFAE  init  clr     latche   set to bank a
117 C103 86   61          lda     #'a
118 C105 B7   CC4E        sta     plus
119 C108 8E   FE07        ldx     #ststab  init stack table
120 C10B 10EF 84          sts     0,x
```

```
121 C10E 10EF 02          sts     2,x
122 C111 10EF 04          sts     4,x
123 C114 7E   CD03  theEnd jmp    WARMS
124
125                   end     init
```

Name MEMEND.CMD

Syntax MEMEND (addr) where (addr) is a value from 0 to BFFF and is optional. In both cases the current value of MEMEND is reported.

Listing 2:

```
0                    nam  MEMEND.CMD
1                    * Memory end report and set
2
3     CC2B  MEMEND  equ  $CC2B   FLEX memory end pointer
4     CD03  WARMS   equ  $CD03   FLEX return address
5     CD24  PCRLF   equ  $CD24   cr and lf
6     CD42  GETHEX  equ  $CD42   FLEX get hex number routine
7     CD45  OUTADR  equ  $CD45   output 2 hex numbers
8
9     C100          org  $c100
10    C100 BD CD42  start jsr  GETHEX  get number to set memend to
11    C103 25 0B C110      blo  report  no change
12    C105 5D             tstb
13    C106 27 08 C110      beq  report
14    C108 8C BFFF        cmpx ##$BFFF  Check for high bounds
15    C10B 22 03 C110      bhi  report
16    C10D BF CC2B        stx  MEMEND  set new memend
17
18    C110 8E CC2B  report ldx  #MEMEND
19    C113 BD CD24        jsr  PCRLF
20    C116 BD CD45        jsr  OUTADR
21    C119 BD CD24        jsr  PCRLF
22    C11C 7E CD03        jmp  WARMS
23
24                   end  start
```

Notes

The BANK program breaks some rules and does some things in a crude way. I will use the excuse that I wanted to keep it simple for the magazine's sake, and yours, in typing it in. The truth of the matter is that when the program started working, I stopped. The user command table in FLEX can only be extended once. It already has the 'P' command added to it. That is why I included the 'P' command in the table in this program.

The correct way would have been for the program to do this rather than the brute force method I used. If you run programs that add to the UCT, then you should run them after you run this program. Programs that add to the UCT are MCOMMAND, AUTOTASK and JCP, among others. I run the BANK command from the STARTUP file so this is never a problem. When you switch to a bank, the stack pointer

is saved and the environment is pushed on the stack. Then the stack pointer for the bank we are going to use is restored, and the environment is pulled from that stack.

If the stack for both banks is the same, then the environment will be corrupted. This may be a problem with some programs. The proper way would be to save the environment in some protected area of memory. There is enough room here to do that, I just didn't get to it.

The location $FE00 contains the number of the bank we are in, either 0, 1 or 2. This can be tested from your program. It would be possible to switch banks by poking the appropriate locations in memory. If you do this, you should set $FE00 to indicate what you did for other programs to use.

You should also consider what the escape return register and the stack pointer will hold. It might be wise to

use the locations in this program for that. The other use for the extra 64K, as a RAM disk may be ready by the time you read this. Contact DSL for information on that.

As a RAM disk, you could get 256 sectors of programs and data in very fast access. This is another use for the extra memory that is different than what I've done here — RGS update. The system I did this on is an older one, and the 128K upgrade is no problem. The newer CoCo 2 has some space problems, and the 128K board from RGS that I got was an early one and had some minor bugs. I have sent it back to them, so I haven't had a chance to get this program to work with it.

I don't see any problem, though. Just change the code that switches banks to the RGS board, and it should work. I will give you an update, probably next month, about this.

OS-9 UTILITY

Random Numbers For The OS-9 C Compiler

By Lew Middaugh

For want of something or the other, an empire was lost. But without a random number generator, I could not build a fantasy empire to conquer. So I wrote a random number generator.

I wrote an assembly language random number generator for the Color Computer OS-9 C Compiler. It is written to allow its addition to a user C library. This article presents this random number generator and an example of its use in a game program. Along the way I'll briefly describe what a C compiler is, and how you go about preparing an executable program with the C compiler.

How A C Compiler Works,
The Background

A C compiler is nothing more than a program that translates a C language program into a 6809 assembly language program. How is the C Compiler used to prepare a C program? Figure 1 presents an overview of the steps in the process of generating an executable C program.

The first step in preparing a C pro-

(Lewis Middaugh, a systems engineer at GTE, likes computers and FRP games. He is working on a fantasy game and finding this a never-ending quest.)

gram is to enter a C program using an editor. Unlike CoCo BASIC where you can just type in your program, a text editor is required to input your C program for the C compiler. The editor that is provided in the OS-9 operating system package can be used to input a C program. After your C program has been entered, it is then saved as an ASCII text file on the disk.

Once your program is safely stored on the disk, the C compiler is loaded and executed. The C compiler performs the remaining two steps in creating an executable C program.

First, the C compiler reads your program from the disk. Each line of the C program is compiled (translated) into a series of assembly language instructions. These resulting assembly language instructions are stored on the disk during compilation. Example 1 is an annotated listing of how two lines of a C program would be translated into assembly language.

The OS-9 C Compiler performs this translation process from C language to assembly language in four phases: macro pre-processing, compiler pass 1, compiler pass 2, and assembly language code optimization.

Next the OS-9 C Compiler starts the assembler program. An assembler trans-

lates the assembly language program into machine language. The assembler is loaded and executed and reads the assembly language program from the disk. During assembly the resulting executable machine code is stored on the disk.

The OS-9 C Compiler performs this process from assembly language to machine code in two phases — assembly and linking. It is the linking process that examines the C program for references to the user and system C library functions. If a reference is found, that library function is linked or added into the machine code version of the C program.

It is not necessary to completely understand assembly language programming in order to program in the C language. The programming is done in C; assembling the output of the C compiler is just a step in the translation process. However, to be effective in programming in C, it helps to understand assembly language.

Finally the C language program, now in machine code, can be loaded and executed. This whole process generates fairly fast executing machine code, in the ballpark of 10 to 100 times faster than CoCo BASIC. This is fast, but assembly language programming is still a little faster.

In summary, there are three steps in generating an executable C program. The first step is to enter the C program using a text editor. The second step is to use the C compiler to translate the lines of C program into a series of assembly language instructions. And the last step is to assemble these assembly language instructions into the binary machine code that the CoCo understands. This binary machine code is the executable version of the C program that can be loaded and executed.

Yes, it is longer and more involved than typing in a BASIC program. But the results are sometimes well worth the effort.

What Is A Library File?

In order to perform certain functions, the C compiler references a collection of commonly used position independent subroutines. These subroutines are called library routines and are contained in a library file. Routines from this file are added to your program during the assembly process.
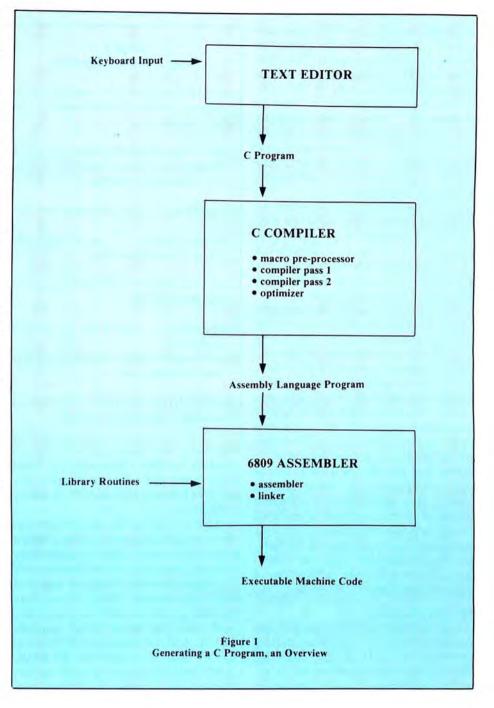
This library file contains the mathematical and logical functions needed to execute your program. These routines include addition, division, logical *AND*, arithmetic shift, and others. It also contains the keyboard, display, and disk I/O routines, character compare routines, and initialization and exit routines. Thus, these assembly language routines provide most of the common functions needed when executing a program. It is around these functions that the compiler builds your programs. Table 1 provides a partial listing of the library routines contained in the OS-9 C library.

If a function is needed that is not in the library, the programmer has at least three options. One way is to write a subroutine in C that performs this function in terms of the available functions. Another way is to simply do without. Or, the programmer could write a new library function. This last approach is how I chose to create a random number generator function.

The Random Number Library Routine

There were two things on my mind when I started. How does one generate a random number? And how does a C program call a machine language function? With a little research and the inspection of the assembly output of the C compiler, these questions were adequately answered.

From the research I found a method of generating random numbers that is

Figure 1
Generating a C Program, an Overview

**Example 1
A Sample of the C Compiler Output (annotated)**

```
*  *      printf("nPLAY AGAIN? (Y,N) n");
   leax _6,pcr    get the memory address of the print string
   pshs x         save it on the system stack as required by printf()
   lbsr printf    go print the string using the library function printf()
   leas 2,s       adjust the system stack pointer to remove the string
                  address


*  *      again = inkey ();
   lbsr inkey     go get the next character from the keyboard using the
                  user library function inkey ()
   stb 0,s        store the new value of 'again'
```

simple and easy. In the *TTL Cookbook* by Don Lancaster, there is a section on pseudo-random sequencers. These are based on shift registers that provide a pseudo-random sequence of ones and zeros. It is pseudo-random in that a 31 bit sequence of ones and zeros repeats itself every two billion bits or so. By grouping these ones and zeros into groups of eight, a number between zero and 255 is generated that is sufficiently random for most game uses.

The second question was answered by examining the assembly language output of the C compiler. C programs pass arguments (variables) to a function on the system stack. The system stack is also used to hold return addresses for subroutine and interrupt calls.

If a value is returned by the function, it is placed in the 16-bit D register by the function and retrieved by the calling program after the function is finished.

These cryptic remarks will be best cleared up by looking at Listing 1. This contains the rnd() routine that generates a random number. It returns an integer value between zero and 255. It is a software implementation of a hardware pseudo-random sequence generator.

However, this routine is not as nice as CoCo's *RND* function in BASIC. Listing 2 is a routine called die(). It returns a number between one and a specified number, up to a maximum of 255. This is similar to the CoCo BASIC *RND* function as it is commonly used. The following paragraphs provide explanation of these two routines.

The rnd() Routine

Line 20 is the label for the random number routine. Its value is the starting address of the rnd() routine. The value of this label is assigned when the library routines are linked to your C program during the translation process to create a machine executable program.

The first thing that needs to be done is initialization. Line 21 causes register X to point to the seed that is used to generate the next random number. This is done using the indexed addressing mode. As register Y points to the beginning of the data area and the offset will be set during the translation process to the location of the seed in the data area, this allows this instruction to be position independent. Thus, the rnd() function can be relocated to a different address and still perform properly without reassembly.

In Line 22, register B is loaded with 8. Register B is used as a counter. When it reaches zero we will have a new 8-bit

random number. (Remember, it takes eight ones and zeros to express any number between zero and 255 in binary notation.)

As every call to this function causes the seed to change, every call will return a different number. Also note that every time the program is loaded into memory the same sequence of random numbers will be generated. Thus, this random number generator suffers a same drawback that the BASIC *RND* function suffers. And, likewise, some of the techniques used to randomize the BASIC *RND* function could be used to randomize the C random number generator.

Starting at the label on Line 23 through Line 30, the task is to set the carry flag based on the EXCLUSIVE-OR of bit 28 and bit 31 of the previously mentioned seed. (We are referring to the seed as if it was a big four-byte 32-bit register where bit 1 is the leftmost bit.) This is the heart of the random number generator.

Lines 31 through 34 rotate the bits of the seed to the right. This causes the contents of the carry flag to be placed into the leftmost bit of the seed as the new most significant bit.

By doing this eight times, as provided by Lines 35 and 36, the eight most significant bits of a new seed have been generated.

These bits are returned as the new random number in register D. Line 37 sets register D for return by loading D with a preset zero and the most significant byte of the random number seed. Line 38 then returns control to the calling routine.

The die() Routine

The die() routine, shown in Listing 2, scales the result of the rnd() routine to the desired range. In this, it works the same way the CoCo BASIC *RND* function works. For example, in CoCo BASIC, *RND(8)* returns a random number between one and eight. For the C language program, die(8) would also return a random number between one and eight.

How is this done? Line 24 defines the starting location of this routine. Line 25 gets a random number into the lower half of register D. (The D register is the A and B registers referenced as one register.) Line 26 loads register A with the number of sides on the die from the system stack. This number was saved on the user stack by the calling routine. The random number and the number of sides on the die are multiplied in Line 27.

To complete the scaling of the number, we need to divide the result by one more than the maximum value of the random number. This number is 256, which is also conveniently the same thing as dividing the result by two, eight times. Or by simply using the highest byte of the multiplication result in register A and ignoring the lowest byte in register B, we have effectively divided by 256. This is done in Lines 28 and 29.

The number in register B ranges from zero to one less than the number of sides of the die. Line 30 corrects this by adding one to this number in register B. Now the die roll is in register D, where the C program expects to find it on return. Then in Line 31, with its work done, the die() function returns control to the C program.

Example C Program

Three-Roll is a dice game that uses the die() routine. See Listing 3. This game is fairly simple at first glance. It is you vs. the house, which is your CoCo. And it's a tough game, with the probability entirely in favor of the house.

You get to roll a die four times adding the rolls to form a total. The house is also rolling a die, forming the house total. After each roll, you may raise your bet, which is matched by the house. After the fourth roll, if your total

is higher than the house, you win the game and the pot.

However, there is one little complication. If you get six or more ahead of the house after any roll, you lose the game (and the pot!).

One small detail includes the following. In order to play, you must ante up a small amount in order to play each game of *Three-Roll*.

Three-Roll

Like PASCAL, C is also a structured programming language. In this it lends itself easily to a modular approach of program design. Structure charts can be used to help document this modularity and the interrelationships between the modules. Figure 2 presents a structure chart for the *Three-Roll* program.

As you can see, the *Three-Roll* program is broken up into five modules. The die() function used by the module roll is not shown. This function is a library function. The library functions used by *Three-Roll* are not shown.

The main module oversees the entire series of *Three-Roll* games that the user wishes to play. It initializes the number of tokens the player has to bet and determines if the the player wishes to

play again. Also, if the player is low on tokens, the house (CoCo) is happy to extend credit!

The randomize module asks for an input from the keyboard. Using this input, this module calls the rnd() library function the specified number of times in order to randomize the sequence of die rolls. This is not necessarily the best technique to randomize the rnd() function, but it is usable.

The playgame module plays a single game of *Three-Roll*. It causes the new die totals to be rolled, gets the player's bets, and determines if the player (you) either won or lost. It then returns to the number of tokens the player now has to the main module.

The roll module rolls the die for the house and the player. It sums up the new totals and informs the player of the rolls and the new totals. When this is done, it returns to the playgame module with the new house and player totals.

The getbet module asks the player for his bet. If the bet is not valid, for example, if the bet is for more tokens than the player has, the player is told the number of tokens left and is asked again for a bet. After a valid bet has been entered, the module returns to playgame with

the bet.
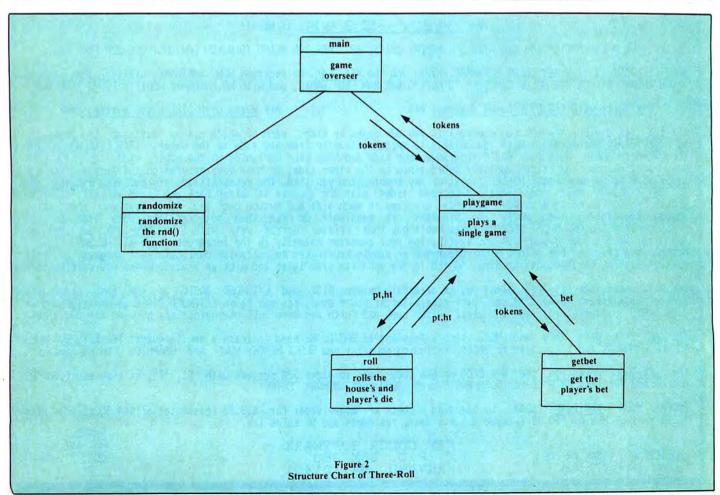
The program, *Three-Roll*, is shown in Listing 3. Using the module descriptions, most of the program lines can be followed. For further details on the C language, refer to one of the two C program language texts listed in the references.

A Loose End, The Inkey() Function

In the program listing of *Three-Roll* a function called inkey(), is used. This function is not a C function, but a library function that is included in my user library along with the die() and rnd() functions. It is not shown in the structure chart of Figure 2 as it is a library function. This function is shown in Listing 4.

Normally the C library function, getc(), or its derivative, getchar(), is used to get character input from the keyboard. This function is a line buffered read from the keyboard that returns a single character.

Thus, on the first call to getchar(), a line is read in from the keyboard. This line consists of all the characters that you typed in before pressing the ENTER key. The getchar() returns only the first character entered.

Figure 2
Structure Chart of Three-Roll

On the next call to getchar(), the second character that you entered is returned. This continues until all the characters you entered (including the ENTER key) have been returned. On the next call, another line is read in from the keyboard and the first character of the line is returned. And now the process continues as before.

What I wanted was an unbuffered, single-character read from the keyboard. In short, I wanted an *INKEY$* function somewhat like CoCo BASIC has. The inkey() function returns the character value of the next key pressed on the keyboard.

What have we accomplished? We now have a random number generator for the OS-9 C Compiler; fantasy monsters beware! You have also read an introduction on how to use a C compiler and to the concept of a library function. (In many ways, these library functions are similar to BASIC's built-in functions.) As an example of the use of this random number generator, a dice game was presented. (A perfect pastime for fantasy Adventurers relaxing in the tavern.)

References

The following references were used in understanding the C language and the application presented in this article.

Hancock, Les, *The C Primer, (A Byte Book)*, McGraw-Hill Book Company, 1982. This book is intended for programmers who know nothing about the C language. This is not a complete description of the C language, but

> *"Like PASCAL, C is also a structured programming language. In this it lends itself easily to a modular approach of program design. Structure charts can be used to help document this modularity and the interrelationships between the modules."*

provides the "stripped-to-essentials outline of C a beginner needs."

Kernighan, Brian W., and Ritchie, Dennis M., *The C Programming Language*, Prentice-Hall, Inc., 1978. The standard reference on the C programming language.

Lancaster, Don, *TTL Cookbook*, Howard W. Sams and Co., Inc., 1974, pages 277-281. This book describes the hardware pseudo-random sequencer on which the random number generator routine is based.

Leventhal, Lance A., *6809 Assembly Language Programming*, Osborne/McGraw-Hill, Inc., 1981. One of the earliest reference books on 6809 assembly language programming. And still one of the best.

Ley, B. James, *Computer Aided Analysis and Design for Electrical Engineers*, Holt, Rinehart, and Winston, Inc., 1970. Chapter 10 on random number generators contains a discussion of random number generators and tests that can be used to verify that a random number generator is such. Using these tests, I satisfied myself that this random number generator was adequate for game use. Further testing would be required if it is to be used for any serious application.

Listing 1:

```
00001 * rnd.a - random number generator  Ver 1.0
00002 *    Copyright (c) 1984 by L.A.Middaugh
00003 *    written to be included in a user library
00004 *    for RS/Microware's CoCo OS-9 C Compiler, Ver 01.00.00
00005 *    Revised version of RND,  11-82 LAM
00006 *    written for Duggar's Small C Compiler, Ver 1.0
00007 *
00008 * Purpose: rnd.c generates a random number between 0 and 255.
00009 * Call Sequence:  rn = rnd()
00010 *    rn = value returned by rnd() function
00011 * Initial conditions:  initial seed at label 'seed'
00012 * Registers affected: A,B,X
00013 * Results:  value returned in register D.
00014 *
00015
00016                    psect rnd_a,0,0,0,0,0
00017 0000              nam   rnd_a
00018 0000              ttl   rnd
00019
00020 0000     rnd:
00021 0000 30a90000     leax  seed,y  point to seed
00022 0004 c608         ldb   #8      8 bits per random number
00023 0006     loop
00024 0006 a604         lda   4,x     exclusive or
00025 0008 46           rora          bit 28 with bit 31
00026 0009 46           rora
00027 000a 46           rora
00028 000b a804         eora  4,x
00029 000d 46           rora          place result
00030 000e 46           rora            in carry flag
00031 000f 6601         ror   1,x     rotate carry into bit 0
00032 0011 6602         ror   2,x       as rotating entire
00033 0013 6603         ror   3,x       seed right
00034 0015 6604         ror   4,x
00035 0017 5a           decb          do this
00036 0018 26ec         bne   loop    eight times for new number
00037 001a ec84         ldd   0,x     get new random number
00038 001c 39           rts             and return
00039
00040 * set up a static variable for the seed
00041 0000              vsect
00042 0000 00     seed  fcb   $0      zero for top half of D reg.
00043 0001 52414e44     fcc   "RAND"  the seed
00044 0005              endsect
00045
00046 001d              endsect
```

Listing 2:

```
00001 * die.a - die thrower  Ver 1.0
00002 *    Copyright (c) 1984 by L.A.Middaugh
00003 *    written to be included in a user library
00004 *    for RS/Microware's CoCo OS-9 C Compiler,  Ver 01.00.00
00005 *    Revised version of DIE,  11-82 LAM
00006 *    written for Duggar's Small C Compiler, Ver 1.0
00007 *
00008 * Purpose: die.a generates a random number between
00009 *    1 and the number of sides on the die.
00010 *    The maximum number of sides is 255.
00011 * Call sequence: rn = die(sides)
00012 *    rn = value of dice roll returned
00013 *    sides = number of sides on die
00014 * Initial condition: second double-byte on system
00015 *    stack contains the number of sides on the die.
00016 * Registers affected: A,B,X
00017 * Results: value returned in register D
00018 *
00019
00020                    psect die_a,0,0,0,0,0
00021 0000              nam   die_a
00022 0000              ttl   die
00023
00024 0000     die:
00025 0000=17fffd        lbsr  rnd    get random number into register D
00026 0003 a663          lda   3,s    get # of sides into accumulator A
```

```
00027 0005 3d           mul           now multiply them
00028 0006 1f89         tfr   a,b     get result divided by 256
00029 0008 4f           clra
00030 0009 5c           incb          set 1 as lowest number
00031 000a 39           rts
00032
00033 000b              endsect
00034
```

Listing 3:

```
1  /*------------------------------------------------
2     THREE-ROLL, the dice game
3        Copyright (c) 1983 by L.A.Middaugh
4        Version 2.0,  last revised  5 May 1983
5        revised for OS-9,  30 May 1984
6     ------------------------------------------------*/
7  #include <stdio.h>
8  #define INITIAL_PURSE 100
9  #define ANTE 1
10 #define OR ||
11 #define AND &&
12
13 int ht,pt;       /* house total, player total */
14
15 main()
16   {
17   int tokens; /* number of tokens player has   */
18   char again; /* play again flag               */
19
20   randomize();
```

```
21    again = 'Y';                                          49      nroll;  /* number of rolls this round    */
22    tokens = INITIAL_PURSE;                               50
23    printf("\nYOU START WITH %d TOKENS.\n",INITIAL_PURSE); 51   printf("\nTHE ANTE IS %d.   THANK YOU!\n",ANTE);
24    while ( again=='Y' AND tokens>0 )                      52   nroll=pot = 1;
25      (                                                    53   tokens = tokens - ANTE;
26      tokens = playgame(tokens);                           54   ht=pt=diff = 0;
27      printf("\nYOU NOW HAVE %d TOKENS.\n",tokens);         55   roll();
28      printf("\nPLAY AGAIN? (Y,N) \n");                     56   while ( nroll<4 AND diff<6 )
29      again = inkey();                                     57     (
30      if ( again == 'y' )  again = 'Y';                     58     bet = getbet(tokens);
31      if ( tokens<=1 AND again=='Y' )                       59     pot = pot + bet + bet;
32        (                                                  60     tokens = tokens - bet;
33        printf("\n\nYOU CREDIT HAS BEEN EXTENDED.\n\n");    61     printf("THE CURRENT POT IS %d.\n",pot);
34        tokens = 50;                                        62     roll();
35        )                                                  63     nroll++;
36      )                                                    64     diff = pt - ht;
37    printf("\nTHANKS FOR PLAYING!\n");                      65     )
38    }                                                      66   if ( pt>ht AND diff<6 )
39                                                           67     (
40  /*---------------------------------------------          68     tokens = tokens + pot;
41     Play a game routine,  4 or less die rolls             69     printf("\nYOUR TOTAL IS GREATER\n   THAN THE HOUSE.");
42     ---------------------------------------*/             70     printf("\nYOU WIN !\n");
43  playgame(tokens)                                         71     )
44  int tokens;                                              72   else
45    (                                                      73     (
46    int pot,     /* total number of tokens in pot */       74     if ( diff>=6 )
47        diff,    /* difference in totals         */         75       printf("\nYOUR TOTAL IS 6 OR MORE GREATER");
48        bet,     /* the player's bet             */         76     else printf("\nYOUR TOTAL IS NOT GREATER");
                                                             77     printf("\n   THAN THE HOUSE.");
                                                             78     printf("\nYOU LOSE...\n");
                                                             79     )
                                                             80   return(tokens);
                                                             81   }
                                                             82
                                                             83  /*---------------------------------------------
                                                             84     Get a valid bet from the player
                                                             85     ---------------------------------------*/
                                                             86  getbet(tokens)
                                                             87  int tokens;
                                                             88    (
                                                             89    int bet;      /*  the player's bet          */
                                                             90
                                                             91    bet = -1;
                                                             92    while ( bet < 0 )
                                                             93      (
                                                             94      printf("\nYOU HAVE %d TOKENS.\n",tokens);
                                                             95      printf("WHAT IS YOUR BET? (0-9) \n");
                                                             96      bet = inkey() - 48;
                                                             97      printf("\n");
                                                             98      if ( bet>9 OR bet<0 )
                                                             99        (
                                                            100        printf("   THAT BET IS NOT PERMITTED.\n");
                                                            101        bet = -1;
                                                            102        )
                                                            103      if ( bet > tokens )
                                                            104        (
                                                            105        printf("   BUT YOU ONLY HAVE %d TOKENS.\n",tokens);
                                                            106        bet = -1 ;
```

```
107        }
108     }
109  return(bet);
110  }
111
112  /*---------------------------------------------
113     Randomize the rnd() function.
114     ---------------------------------------*/
115  randomize()
116  {
117  int count,   /* Number of times to call rnd() */
118      dummy;   /* A place to dump rnd() result  */
119
120  printf("\nHIT ANY KEY TO START.\n");
121  count = inkey();
122  while ( count-- > 0 )   dummy = rnd();
123  return;
124  }
125
126  /*---------------------------------------------
127     Roll the dice and total results function
128     ---------------------------------------*/
129  roll()
130  {
131  int hr,pr;  /* house's roll,  player's roll */
132
133  hr = die(6);  pr = die(6);
134  ht = ht + hr;  pt = pt + pr;
135  printf("HOUSE ROLLS A %d\n",hr);
136  printf("YOU  ROLL  A  %d\n",pr);
137  printf("\n HOUSE TOTAL IS NOW %d\n",ht);
138  printf("  YOUR  TOTAL IS NOW %d\n",pt);
139  return;
140  }
141
```

Listing 4:

```
.es 69
.es 71
.nf
.br
    1  /*    Unbuffered single character input
    2       (see Kernighan & Ritchie, page 161)
    3       works much the same as CoCo BASIC INKEY$
    4       except it waits for next key pressed.
    5  */
    6  #include <stdio.h>
    7  #define CMASK   0377 /* for making char's >0 */
    8
    9  inkey()
   10  {
   11  char c;
   12  return( (read(0,&c,1) > 0) ? c & CMASK : EOF );
   13  }
```

RANDOM BASICS

Design And Development Of Application Software — Part V

By Paul Searby
Rainbow Contributing Editor

This month's article is the last in the series on design and development of application software. There could be more, but I feel that I have accomplished my goal of describing and defining a method that will assist someone involved in developing a large application system. One of the main points I hope has come across is that organization of your thoughts is critical to the success of your project. Flowchart (outline) your system before thinking about the programming. Define your file layouts and report descriptions before you start the programming. As you go through these steps, you will think of items that should be considered and included before they become pro-

(Paul Searby has been involved with computers and data processing since 1969, working primarily on larger IBM systems until 1975 when he bought, built and programmed the first "personal computer" ever made — an Altair 8800, which came in kit form with 1 K of memory. In 1977 he left a position as project manager for a large corporation to devote his full efforts to his company, Computerware.)

gramming changes. Within these articles, Random BASIC was the language selected for the programming, but with many other high level languages now available for the CoCo, another language may be more suited to your particular application.

It is best to complete the design, set it down for a day or two, then come back to it and make the decisions as to language and the order in which the programs should be developed. We have talked about using "Top Down" methods in our development, including the programming phase. With the same qualification I gave to doing structured programming in BASIC, we will borrow from the Top Down programming philosophy, but not strictly adhere to it. This means that we will program some of the higher levels first, but when we get to a point that we can't continue writing code that can be tested, we will succumb to the lower levels and write the code required to keep progressing at the higher levels.

Did anyone guess what the scope of our Accounts Payable System really was? Since this is the last of the articles, I will give you my best estimate of the answer. At a minimum, it is nine pro-

grams, with an estimated total size of 75K to 90K worth of code. My estimate is based on having seven different completed application systems to compare. We could write one that takes less space or possibly one or two less programs, but our original charter included both user friendliness and maintainability. To keep those and our other given guidelines, the above estimate is probably, at best, conservative. Following is a breakdown on the nine proposed programs:

1) APS100.BAS — Master File Processing (Chart 2)
2) APS120.BAS — Keyfile Generation / Reorganization
3) APS200.BAS — Invoice File Processing (Chart 3)
4) APS400.BAS — New Period Processing (Chart 4)
5) APS500.BAS — Report Requests (Chart 5)
6) APS520.BAS — Report Printing (Chart 5)
7) APS900.BAS — Utility Processing and Master Menu (Chart 6)
8) SYS100.BAS — Generalized Sort (Chart 5)

9) SYS120.BAS — Generalized Merge (Chart 5)

Several of the programs have not been mentioned much because they are what I term "system programs." They are typically included in every system, and thus do not have to be written each time. In Accounts Payable, we will have a need for the generalized sort and merge programs. These utilities are used for sorting and merging extracted files for either report or keyfile generation purposes. They can be used in any application because the calling program tells them where to return control after they are finished. Another remaining program borders on being a general purpose utility: the *Keyfile Generation and Reorganization* program. The ability to generate and reorganize the *Master's Keyfile* is required in any system that uses Keyfile access. Thus if we

knew for sure that we were going to write additional applications using keyfiles, we might want to take the extra time to code *APS120.BAS* a little more general and place it in our library of SYSxxx programs.

The remaining six programs pretty much follow our flowchart. As to where to start, my choice would be parts of *APS900.BAS* (see examples), which has our top level Master Menu and many of the utilities required to get our system off the ground. Initially, I would only code the Master Menu, the System Control File creation and maintenance, the date routine and the Master, Invoice, Activity and Keyfile creation routines. The other features of *APS900* can be added at a later date. After getting this working, we can now branch out into the other programs. Next would be *Master File Processing*, since all other files are dependent on the *Master File*.

Since access to the *Master* is via the *Keyfile*, we pretty much need to develop our *Master* and *Keyfile* logic together. This is where it gets nice to already have a *Keyfile Generation and Reorganization* utility. Assuming this is the first time, we will need to develop this logic as we work on the *Master* and *Keyfile* sections.

With a system that will end up being 75K to 90K in size, it is not easy to choose which examples of the coding should be included. I have chosen the "front end," being the Master Menu portion of the utility program and most of the file I/O logic of the *Vendor File Maintenance* Program. Also included is a generalized error routine that is appended to the end of each program. When you study these examples, please remember that they are only excerpts from programs, and as such have loose ends.

```
---> Master Menu Example <---
.sp2
0100 : APS900.BAS  -  A/P Master
 Menu and Utilites
0120 : (C) 1984 Computerware - V
ersion 1.0  09/25/84
0140 :
0160 LINE= 0 : STRING=24 : DIGIT
S=0 : RJUST=0
0180 ON ERROR G.9999
0200 LET Z$="APSSYS.DTA" : OPEN\
#19,Z$
0220 READ \#19,PTR,CRT,DATE,DRV1
,DRV2,DRV3,XX,OPT1,OPT2,CNAME$
0240 CLOSE \#19
0240 :
0280 LET A$=STR$(DATE) : IF DATE
 < 750000 W$=" " : G.340
0300 LET W$=MID$(A$,3,2)+"/"+MID
$(A$,5,2)+"/"+MID$(A$,1,2)
0320 :
0340 HOME
0360 LET T0=24-LEN(CNAME$)/2 : P
.TAB(T0);CNAME$
0380 PRINT TAB(12);"Accounts Pay
able System" : P.
0400 PRINT TAB(12);"System Date
is : ";W$ : P.
0420 PRINT TAB(05);"All Transact
ions Entered will use this Date"
 : P.
0440 PRINT TAB(8);" 0  Return to
 the System Menu" : P.
0460 PRINT TAB(8);" 1  Vendor Ma
ster Processing"
0480 PRINT TAB(8);" 2  Receive /
 Pay Invoices"
0500 PRINT TAB(8);" 3  New Perio
d Processing"
0520 PRINT TAB(8);" 4  Reserved"
0540 PRINT TAB(8);" 5  Report Re
quest Processing"
0560 PRINT TAB(8);" 6  Reserved"
0580 PRINT TAB(8);" 7  Reserved"
0600 PRINT TAB(8);" 8  Change th
e System Date"
0620 PRINT TAB(8);" 9  Utility P
rocessing" : P.
0640 PRINT : P.TAB(9);
0660 INPUT "Please make a select
ion...",A : P.
0680 IF A=0 C."SYS000.BAS"
0700 IF A=1 C."APS100.BAS"
0720 IF A=2 C."APS200.BAS"
0740 IF A=3 C."APS400.BAS"
0760 IF A=5 C."APS500.BAS"
0780 IF A=8 G.860
0800 IF A=9 G.1160
0820 GOTO 640
0830 :
0840 : Date Input Routine
0850 :
0860 PRINT : INPUT"Enter Month (
1 - 12)",MTH
0880 IF MTH < 1 G.860
0900 IF MTH > 12 G.860
0920 INPUT "Enter Day (1 - 31)",
DAY
0940 IF DAY < 1 G.920
0960 IF DAY > 31 G.920
0980 INPUT "Enter Year (19xx)",Y
R
1000 IF YR < 78 G.980
```

```
1020 IF YR > 99 G.980
1040 LET DATE=YR*10000+MTH*100+D
AY
1060 :
1080 LET Z$ = "APSSYS.DTA" : OPE
N\#19,Z$
1100 PUT \#19,PTR,CRT,DATE,DRV1,
DRV2,DRV3,XX,OPT1,OPT2,CNAME$
1120 CLOSE \#19 : G.300
1140 :
1160 : System Utilites Start her
e.....
1180 :
.PG
.sp
.ce
---> Vendor File Maintenance Rou
tines <---
.sp2
1000 : The folowing are routines
 out of the Vendor File Maint.
1020 : Program.  They include th
e Master File Read and Write,
1040 : along with all the Keyfil
e Search routines.  Also include
d
1060 : are the Keyfile Extract f
or a new Keyfile and the Error
1080 : routine that is common to
 each program in the System.
1100 :
1120 : As an example, to get a V
endor via the ISAM Binary Search
:
1140 :
1160 INPUT "Please enter the des
ired Vendor Name",TARG$
1180 LET LT = LEN(TARG$) : GOS.2
260 : IF HIT <> 0 G.1260
1200 PRINT : INPUT "Do you want
to try again (Y/N)",ANS$ :P.
1220 IF ANS$ = "Y" G.1160 :ELSE
exit to Menu
1240 :
1260 RECNO\#10 = KLOC : GOS.1440
 :: You now have Master!
1280 :
1300 : The Vendor Master output
(write) routine can handle both
1320 : Adds and Changes.  For an
 Add, MODESW = 1 - for a Change,
1340 : MODESW = 2.  MODESW is ac
tually the selection option from
1360 : the Vendor File Maintenan
ce Menu, serving dual purposes.
1380 :
1400 : Read Master
1420 :
1440 GET \#10,VIDX,VNUM,STINV,LS

TINV,SPARE,BEGBAL,CURIN,CURPD,YT
DPD :
1460 : LSTDT,NAME$,TITL$,ADDR$,C
TST$,PHAC,PHNUM,TERMS
1480 LET NAMSV$ = NAME$ : RET.
:: Save Vendor Name (KEY)
1500 :
1520 : Vendor Master Output Rout
ine
1540 : Find space for Master add
 in Keyfile
1560 : (VSTRT initially = start
of the Keyfile)
1580 : (Deleted Master entries h
ave their pointer negated)
1600 :
1620 IF MODESW <> 1 G.1900
1640 RECNO \#13 = VSTRT
1660 IF RECNO\#13 >= RSIZE\#13 G
.2040
1680 IF RECNO\#13 >= RNEXT\#13 R
ECNO\#10 = RNEXT\#10 : G.1780
1700 GET \#13,TEST : IF TEST >=
0 VSTRT = VSTRT+1 : G.1640
1720 : Found a spot (TEST was ne
gative)
1740 LET LOC = ABS(TEST) : RECNO
\#10 = LOC : TEST = 0
1760 PUT \#13,TEST : VSTRT = VST
RT+1
1780 LET LOC = RECNO \#10
1800 RECNO \#13 = RNEXT \#13
1820 PUT \#13,LOC,NAME$ : LIDX =
 LOC
1840 :
1860 : Write out Vendor Master
1880 :
1900 PUT \#10,VIDX,VNUM,STINV,LS
TINV,SPARE,BEGBAL,CURIN,CURPD,YT
DPD :
1920 : LSTDT,NAME$,TITL$,ADDR$,C
TST$,PHAC,PHNUM,TERMS
1940 :
1960 IF MODESW <> 2 RET. :: 2 =
change (was Key chgd?)
1980 IF NAME$ = NAMSV$ RET.
2000 PUT \#13,0 : CURPOS = RECNO
\#13 : RECNO\#13 = RNEXT\#13
2020 PUT \#13,KLOC,NAME$ : RECNO
\#13 = CURPOS : RET.
2040 :
2040 HOME : P."KEYFILE FULL - yo
u MUST reorganize the Keyfile be
fore"
2080 PRINT "you can continue usi
ng the Accounts Payable System..
"
2100 : [ here would be a GOTO to
 the Vendor Maintenance Menu ]
```

```
.PG
.SP2
2120 :
2140 : Binary Search Routine
2160 :
2180 : Entry Variables are TARG$
 and LT ( = LEN of TARG$)
2200 : OVFL = Start of Overflow
Area (in 1st Vendor Record)
2220 : Output is KLOC (Record \#
 of Master) and HIT (Ø = no)
2240 :
2260 LET HIT = Ø : FIRST = 6 : L
AST = OVFL-1 : OVFLG = Ø
2280 LET MID = INT((FIRST+LAST)
/ 2)
2300 RECNO \#13 = MID
2320 GOSUB 3240
2340 IF TARG$ = KEY$ G.2420
2360 IF TARG$ > KEY$ IF MID < LA
ST THEN FIRST = MID+1 : G.2280
2380 IF TARG$ < KEY$ IF MID > FI
RST THEN LAST = MID-1 : G.2280
2400 GOTO 2860
2420 LET CURPOS = RECNO \#13 : I
F CURPOS = FIRST G.2520
2440 RECNO \#13 = CURPOS-1
2460 GOSUB 3240

2480 IF TARG$ = KEY$ G.2420
2500 RECNO \#13 = CURPOS : GOS.3
240
2520 IF KLOC <= Ø GOS.3200 : G.2
520
2540 IF TARG$ = KEY$ G.2620 : EL
SE G.2860
2560 :
2580 : Got a Hit - Exit
2600 :
2620 LET HIT = 1 : RET.
2640 :
2660 : Re-entry to look for more
 with same Key
2680 :
2700 IF OVFLG > Ø G.2900
2720 GOSUB 3200
2740 IF TARG$ <> KEY$ G.2860
2760 IF LEN(VENKEY$) < 2 GOS.320
Ø : ELSE G.2620
2780 GOTO 2740
2800 :
2820 : Sequential Search thru ov
erflow Area
2840 :
2860 IF OVFL >= RNEXT \#13 G.304
Ø
2880 RECNO \#13 = OVFL-1 : OVFLG
```

```
        = 1
 2900 GOSUB 3200
 2920 IF KLOC <= Ø G.3020
 2940 IF TARG$ <> KEY$ G.3020
 2960 IF LEN(VENKEY$) < 2 GOS.320
 Ø : ELSE G.2620
 2980 GOTO 2940
 3000 :
 3020 IF RECNO \#13 < RNEXT \#13-
 1 G.2900
 3040 IF HIT > Ø G.3080
 3060 PRINT : P.TARG$;" was not f
 ound....." : G.3100
 3080 PRINT : P."That's all I hav
 e!!" : P.
 3100 INPUT "Press RETURN to cont
 inue",A
 3120 LET HIT = Ø : RET.
 3140 :
 3160 : Read a Record from the Ke
 y file
 3180 :
 3200 IF RECNO\#13 >= RNEXT\#13-1
  KEY$ = "" : RET.
 3220 RECNO \#13 = RECNO \#13+1
 3240 GET \#13,KLOC,VENKEY$
 3260 LET KEY$ = LEFT$(VENKEY$,LT
 )
```

```
 3280 RETURN
 .PG
 .SP2
 3300 :
 3320 : Keyfile Extract for New K
 ey Generation
 3340 :
 3360 LET Z$ = "SYSWKS.UKY" : OPE
 N\#1,Z$ : SCRATCH\#1
 3380 RECNO \#10 = 1
 3400 RECNO \#10 = RECNO\#10 + 1
 3420 IF RECNO\#10 >= RNEXT\#10 G
 .3560
 3440 IF RECNO\#10 = RSIZE\#10 G.
 3560
 3460 GOSUB 1440 :: Read Master F
 ile
 3480 LET ZLOC = RECNO\#10
 3500 IF LIDX = Ø THEN ZLOC = -ZL
 OC  : : SHOW AS EMPTY
 3520 WRITE \#1,ZLOC,NAME$
 3540 GOTO 3400
 3560 CLOSE \#10,\#1
 3580 LET Z$ = "SYSWKS.CTL" : OPE
 N\#1,Z$ : SCRATCH\#1
 3600 WRITE \#1,"APS120.BAS",1,Ø,
 Ø,Ø : CLOSE\#1
 3620 CHAIN "SYS100.BAS"
 3640 :
 3660 : Generalized Error Routine
 3680 :
 3700 IF ERCODE <> 33 G.3740
 3720 PRINT "File / Program ";Z$;
 " was NOT found !!"
 3740 LIST \#PTR,100-140 : P.\#PT
 R
 3760 PRINT \#PTR,"An unrecoverab
 le ERROR has been detected."
 3780 PRINT \#PTR,"Please save th
 e following information : " : P.
 \#PTR
 3800 PRINT \#PTR,"The ERROR code
  was : ";ERCODE
 3820 PRINT \#PTR,"The Line Numbe
 r was ";ERLINE : P.\#PTR
 3840 IF ERCODE = 33 G.3880
 3860 IF ERCODE > 60 G.3880 : ELS
 E P.\#PTR,CHR$(12) : P.CHR$(7) :
  STOP
 3880 PRINT \#PTR,"The error rela
 tes to disk data file operation.
 "
 3900 PRINT \#PTR,"You may be abl
 e to solve the problem with one
 of the"
 3920 PRINT \#PTR,"System Utiliti
 es - going to Utility Processing
 ..."
 3940 CHAIN "APS900.BAS"
```

After we have a functioning *Master* and *Keyfile,* we can progress to the coding of receiving or paying invoices (Invoice File Processing). This is where a lot of our common routines come into use. We will need to develop the *Invoice File* linkage to the *Vendor Master,* but will be able to reuse the *ISAM Keyfile* lookup for the *Master File* that was developed in *APS100.BAS.* Again, I would concentrate initially on just getting the system capable of reading and writing invoice records, leaving check writers and other frills to another time. This program also needs the capability of writing out activity records. This code should be done as you are doing the adjustment and payment portions.

To really be able to check the integrity of the Accounts Payable System, we need "eyes" or reports. Thus, the next programs to tackle should be the report extract and a crude version of the report printing. Initially, these can be used to insure that the *Master, Invoice* and *Activity* files are getting updated properly. When I say a crude version, I mean that we are trying to implement the reports as designed, but aren't too concerned about tab settings, etc. At this point we primarily want to see the data,

and can make it look prettier once we're convinced that all our update logic is working correctly. Lastly, *New Period Processing* can be coded, allowing us to reset the system and again recheck our inputs.

"It is best to complete the design, set it down for a day or two, then come back to it and make the decisions as to language and the order in which the programs should be developed."

Once we have the BASIC system operational, we can then go back and complete the remaining functions. Possibly by this time, either we have seen a need for some changes, or have shown the system to a potential customer and they have asked for some changes. If we get to this step before the complete system

is programmed, the odds are that less code will have to be changed than if we wait until we are through. An item that is required, but has not been mentioned to date, is a manual. When is the best time to write the manual? Ideally, it should have been written before we started and used as a guideline while programming. In practice, it typically is written after the system is completed. A good compromise, and one that has some benefits, is to develop it while the programming is taking place. If possible, have someone else do the manual. I say this because first, programmers don't generally write good manuals (and they usually hate doing it), and secondly, if someone else is trying to explain the system, they will spot oversights that the programmer may miss. This may not be a program error as such, but more likely a clearer way to present the information.

I am looking for suggestions for subject matter to cover in future articles. The topics should be somewhat general in nature, leaving the specifics, on OS-9 as an example, to those who have columns on those subjects. If you have a suggestion, please send it to me care of Computerware, Box 668, Encinitas, CA 92024.

THESE FINE STORES CARRY THE RAINBOW

The retail stores listed below carry THE RAINBOW on a regular basis and may have other products of interest to Color Computer users. We suggest you patronize those in your area.

ALABAMA
Birmingham — Jefferson News Co.
Florence — Anderson News Co.
Huntsville — Endicott Software
Madison — Madison Books
Montgomery — Trade 'N' Books

ALASKA
Fairbanks — Electronic World

ARIZONA
Phoenix — The Computer Shop
Home Brew Computers
Tri-Teck Computers
Scottsdale — Data Concepts
Tempe — All Systems Go
Books Etc.
Computer Library
Tucson — Anderson News Co.
Minz Electronics

ARKANSAS
Little Rock — Anderson News Co.

CALIFORNIA
Citrus Heights — Software Plus
Downey — The Floppi Disk
El Cajon — Radio Shack
Exton — Software City
Folsom — Computers, Etc.
Fortuna — R&V Sound
Gretna — The Computer Supply Store
Half Moon Bay — Strawflower Electronics
Hesperia — Dessert Sound, Inc.
Hollywood — Levity Distributors
Livermore — Software Galeria
Lompoc — L&H Electronics Emporium
Los Angeles — Polygon Co.
Milpitas — AVS Computer Systems
Modesto — Software Mart
National City — JARB Software
Willy's Electronics
Pacific Beach — Pro Am Electronics
Palo Alto — Printers, Inc.
Sacramento — Tower Magazine
San Diego — Computer Dimension
The Computer Store
Dimensional Software
Disney's Electronics
Radio Shack
San Francisco — News On 24
Santa Rosa — Sawyer's News, Inc.
Southgate — Color Computing
Stockton — Hardings Way News
Sunnyvale — Computer Literacy

COLORADO
Aurora — Aurora Newsland
Colorado
Springs — Hathaway's Magazines
Westminster — Software City

CONNECTICUT
Danbury — Computer Serv. of Danbury
Monroe — Mackey's
Orange — Software City

DELAWARE
Wilmington — Normar, Inc.—The Smoke Shop

FLORIDA
Boca Raton — Software, Software, Inc.
Clearwater — The Avid Reader
Soenen & Wilmoth Books
Software City
Cocoa — Village Book Shop
Davie — Software Plus More
Ft. Lauderdale — Mike's Electronics Distributor
Software Connection
Jacksonville — The Book Nook
Book Town
Kissimmee — Radio Shack & Elec. Hut
Melbourne — City Newsstand
The Little Store
Miami — The News Rack
Orange Park — Software City
Orlando — The Alamo
Software Unlimited
Panama City — Boyd-Ebert Corp.

Computer Systems Group
Pensacola — Anderson News Co.
Sarasota — Family Computers
South
Pasadena — Poling Place
Stuart — Caribbean Engineering Corp.
Tallahassee — Anderson News Co.
Tampa — Fine Print Bookstore
Software City
Sound Trader & Computer Center

GEORGIA
Atlanta — Chips, Inc.
Guild News Agency
Augusta — Software City
Columbus — Muscogee News Co.
Cummings — Kent Radio Shack
Jesup — Kannon Music
Radio Shack
Marietta — Act One Video
St. Simons
Island — Radio Shack
Toccoa — Martin Music Radio Shack

IDAHO
Blackfoot — F/M Systems Electronics
Moscow — Johnson News Agency

ILLINOIS
Aurora — Kroch's & Brentano's
Champaign — Book Market
Chicago — B. Dalton Booksellers
N. Walbash St.
West Jackson St.
Bob's In Newtown
Bob's News Emporium
Bob's Rogers Park
Book Market
East Cedar
North Cicero
West Diversey
E.B. Garcia & Associates
Guild Books & Periodicals
Kroch's & Brentano's
South Walbash
West Jackson
516 N. Michigan
835 N. Michigan
Parkway Drugs
Parkwest Books
Sandmeyer's Bookstore
Univ. of Chicago Bookstore
Univ. of Illinois Bookstore
Videomat, Inc.
Chillicothe — Book Emporium
Danville — Book Market
Decatur — Book Emporium
K-Mart Plaza
Northgate Mall
DeKalb — Appletree Computers
East Moline — Book Emporium
Evanston — Chicago-Main News
Geneseo — B & J Supply
Kewanee — Book Emporium
Lisle — Book Nook
Newton — Bill's TV Radio Shack
Oak Brook — Kroch's & Brentano's
Oak Park — B.I.E.S. Systems
Kroch's & Brentano's
Paris — Book Emporium
Peoria — Book Emporium
Sheridan Village
Westlake Shopping Center
Book Market
Illinois News Service
Schaumberg — Kroch's & Brentano's
Skokie — Kroch's & Brentano's
Springfield — Book Emporium
Sangamon Center North
Town & Country Shopping Ctr.
Sunnyland — Book Emporium
West Frankfort — Paper Place
Wheeling — North Shore Distributors

INDIANA
Berne — White Cottage Electronics

Garrett — Finn News Agency, Inc.
Greenwood — The Computer Experience
Indianapolis — Bookland, Inc.
Indiana News
Jasper — Computer Store
Elex Mart
Lawrenceburg — Bauer Electronics
Madison — Arco Office Supplies
Marion — Computer Corner
Martinsville — Radio Shack
New Haven — Advanced Color Software
Scottsburg — Radio Shack of Scottsburg
Walbash — Mitting's Electronics

IOWA
Bettendorf — Cosmos Computers
Davenport — Interstate Book Store
Software City

KANSAS
Junction City — III H's Stereo
Topeka — Palmer News, Inc.
Town Crier of Topeka, Inc.
Wichita — Amateur Radio Equipment Co.
Lloyd's Radio

KENTUCKY
Greenville — Williams & Wells, Inc.
Hopkinsville — Hobby Shop
Louisville — The Computer Store
Maysville — Radio Shack
Paducah — Radio Shack
Paintsville — Gus-Stan Enterprises
Pikeville — Gus-Stan Enterprises

LOUISIANA
Baton Rouge — Acme Book Co.
Software Solutions
Crowley — Acadiana Newsstand
Shreveport — Computer SOS
Slidell — Radio Shack 7181

MAINE
Brockton — Voyager Bookstore
South Portland — Portland News Co.
Waterboro — Radio Shack

MARYLAND
Lexington Park — Books, Etc.

MASSACHUSETTS
Brockton — Vogager Bookstore
Cambridge — Ninis Corner, Inc.
Out Of Town News
The Program Store
Danvers — Microcon Computer & Software Store
Fitchburg — Corners Book Shop
Ipswich — Ipswich News
Littleton — Computer Plus
Lynn — North Shore News Co.
Woburn — Microcon Softwarecenters

MICHIGAN
Allen Park — Book Nook, Inc.
Ann Arbor — Community News Center
Brooklyn — Weatherwax Radio Shack
Charlotte — Computer Options
Dearborn — DSL Computer Products
Durand — Robbins Electronics
Fenton — Tri-County Electronics Radio Shack
Flint — "I O"
Kalamazoo — John Rollins
Software City
Lapeer — Computer Corner
User Friendly
Lowell — Curt's Sound & Home Arcade Center
Mt. Clemens — Key Book Shop
Michigan Radio
Mt. Morris — Shop And Save
Muskegon — The Eight Bit Corner
Novi — MI Software Dist., Inc.
Okemos — Software City
Owosso — C/C Computer Systems
Hi-Fi Audio Co.
Perry — Perry Computers
Perry Oil & Gas
Pontiac — Computer Shack
Roseville — New Horizons
Royal Oak — Software City
St. Johns — Clinton Electronics

Southfield	Software City
Sterling Heights	Sterling Book Center
Wyoming	Gerry's Book Co.
MINNESOTA	
Minneapolis	Read-More News
MISSISSIPPI	
Grenada	Stereo Store of Grenada, Inc.
Gulfport	Computerland
MISSOURI	
Kansas City	Midwest CoCo Systems
St. Louis	Book Emporium
	Computer Xchange
	Software Centre
University City	Final Edition
NEBRASKA	
Lincoln	Hobby Town
Omaha	Computers & Components
NEVADA	
Las Vegas	Hurley Electronics
	Software City
NEW HAMPSHIRE	
Peterborough	Radio Shack
Portsmouth	Portsmouth Computers
West Lebanon	Verham News Corp.
NEW JERSEY	
Bergenfield	Software City
Cedar Knolls	Village Computer & Software
Cherry Hill	Software City
Clinton	Micro World II
Flemington	Software City
Lawrenceville	Micro Con Software Center
Linwood	Software City
Marmora	Outpost Radio Shack
Montvale	Software City
Morristown	Software City
Pennsville	Dave's Elect. Radio Shack
River Edge	Software City
Rockaway	Software Station
Summit	Software City
Teaneck	Software City
Wayne	Wayne Software
NEW MEXICO	
Albuquerque	East West Enterprises
	Page One Newsstand
	Salt of the Earth
Los Alamos	Sound Center — Radio Shack
NEW YORK	
Brockport	Lift Bridge Book Shop, Inc.
East Syracuse	The Data'Phile Co.
Elmira Heights	Southern Tier News Co., Inc.
Fairport	Software City
Hudson Falls	G.A. West & Co.
Johnson City	Unicorn Electronics
Mt. Kisco	Software City
New York	Barnes & Noble—Sales Annex
	Coliseum Books
	Eastern Newsstand
	Grand Central Station, Track 37
	200 Park Ave., (Pan Am #1)
	55 Water Street
	World Trade Center #2
	First Stop News
	Idle Hours Bookstore
	International Smoke Shop
	Jonil Smoke
	Penn Book
	State News
	Usercom Systems, Inc.
	Walden Books
	World Wide Media Services
N. White Plains	Software City
Plattsburg	Adirondack Computer Supplies
Rochester	Village Green
	World Wide News
Woodhaven	Spectrum Projects
NORTH CAROLINA	
Aberdeen	King Electronics
	Radio Shack
Charlotte	Newsstand Int'l
	Papers & Paperback
Havlock	Computer Concerns
Hickory	C² Books & Comics
Laurinburg	Radio Shack
Marion	Boomers Rhythm Center
Newland	Alpine Electronic Systems
Raleigh	D.J.'s Book and News
	Softmart
Winston-Salem	K & S News Stand
NORTH DAKOTA	
Fargo	Computer Associates
OHIO	
Canton	Little Professor Book Center
Cincinnati	Cinsoft
Coshocton	Utopia Software
Dayton	Wilke News
Fairborn	News-Readers
Kent	The News Shop
Kenton	T.W. Hogan & Associates
Lakewood	Lakewood International News

Lima	Brunner News Agency
	Edu-Caterers
Mayfield	
Heights	Programs Unlimited
	Software City
Miamisburg	Wilke News
Mount Orab	Mount Orab Radio Shack
Rocky River	Programs Unlimited
Toledo	Leo's Book & Wine Shop
	Reitz Electronics
Westerville	Home Computer Store
OKLAHOMA	
Hobart	Shortgrass Electronics
Oklahoma City	Merit Micro Software
OREGON	
Aloha	B. Color Computer and Software
Hermiston	Lee's TV & Radio Clinic
Portland	80-Plus
PENNSYLVANIA	
Allison Park	Software City
Altoona	Newborn Enterprises
Brookville	Larry's Stereo Shop
Coudersport	Weiss Computers
Greensburg	The Program Store
Huntingdon	
Valley	Software City
Malvern	Personal Software
Philadelphia	City Software Center
	Newsy
Phoenixville	Stevens Radio Shack
Pittsburgh	All-Pro Souveniers
Pleasant Hills	Pitt Computer & Software
Pottstown	Video Programming, Inc.
Tunkhannock	The Donna Comm. Co.
Williamsport	Shade Tree
York	The Computer Center of York
RHODE ISLAND	
Newport	Kelly's Variety
Warwick	Software Connection
SOUTH CAROLINA	
Beaufort	Data Byte Computer Center
Charleston Hts.	Software Haus, Inc.
Greenville	Palmetto News Co.
Hilton Head	Megatron Corporation
Spartanburg	Software City
Union	Fleming's Electronics
TENNESSEE	
Chattanooga	Anderson News Co.
Dickson	Highland Electronics
Knoxville	Anderson News Co.
	First Byte Computer Co.
Memphis	Computer Center
	Software, Inc.
	Stewart Software
Nashville	Campus Computer Corp.
	Mosko's Book Store
Smyrna	Delker Electronics
TEXAS	
Austin	Capitol Microcomputers
Conroe	Crouchet Electronics
Elgin	The Homing Pigeon
Ft. Worth	RFI Electronics
	Software Terminal
Houston	MicroSolutions
Irving	Software Access
Orange	Northway Books & News
Paris	Software Solutions
UTAH	
Murray	Deseret Book
Ogden	Computer City
VIRGINIA	
Gafton	Electronics Marketing
Norfolk	I-O Computers
Richmond	Software City
WASHINGTON	
Bellevue	Software City
Marysville	More Than Games
Mount Lake	
Terrace	Emerald Computer Services
Renton	Data Borne
Seattle	Adams News Co., Inc.
Tacoma	Nybbles 'N Bytes
WEST VIRGINIA	
Lesage	Pioneer Technology
Logan	Stan's Electronics & Radio Shack
Parkersburg	Valley News Service
WISCONSIN	
Appleton	Badger Periodicals
Cudahy	Cudahy News & Hobby
Janesville	Book World
Ladysmith	Electronics, Etc.
Milwaukee	Abacus Micro
	Book Tree
	Booked Solid
	Booked Solid II
	Harvey Schwartz Bookshop
	Univ. of Wisconsin Bookshop
WYOMING	
Casper	The Computer Store

AUSTRALIA:	
SYDNEY	
Kingsford	Paris Radio Electronics
CANADA:	
ALBERTA	
Athabasca	McLeans Ltd.
Banff	Banff Radio Shack
Barrhead	Barrhead Sound 1982 Ltd.
Blairmore	L & K Sports & Music
Brooks	Double "D" A.S.C. Radio Shack
Calgary	Billy's News
	Imperial Computer Ltd.
Camrose	Radio Shack Associated Stores
Claresholm	Radio Shack Associated Stores
Coaldale	Coaldale Sight & Sound
Drayton Valley	Langard Electronics
Edmonton	CMD Micro
	Kelly Software Distributors
Edson	Radio Shack
Fairview	D.N.R. Furniture & TV
Fox Creek	Fox City Color & Sound
Ft. Macleod	Fort Pharmacy
	Radio Shack
Grande Cache	The Stereo Hut
Grande Centre	The Book Nook
Grande Prairie	Northern Computer Service
Hanna	Technics TV Ltd.
Innisfail	L & S Stereo
Lacombe	Tall Pine TV
Leduc	Radio Shack Associated Stores
Lloydminster	Lloyd Radio Shack
Peace River	H & S Music Centre
	Radio Shack Associated Stores
Pincher Creek	Thornton & Son's
Redcliff	Gale Distributing
Red Deer	Computer World
Rimbey	Roy's TV
St. Paul	Tele-Logic
Sundre	Sundre Sound
Taber	Pynewood Sight & Sound
Valleyview	Don's Radio Shop
Vermilon	Photocraft Vermilion Ltd.
Wetaskiwin	Radio Shack
BRITISH COLUMBIA	
Burnaby	Compulit
Campbell	
River	TRS Electronics
Chilliwack	Charles Parker
Courtenay	Bell Radio & TV
	Valley Computers
Ft. St. John	Ken Dawson
Merrit	Merrit Radio Shack
N. Vancouver	Microwest Distributors
Sicamous	Shuswap Electronics
Sidney	Sidney Electronics
Victoria	Datafact
	International Software
	Techworld
Williams Lake	Norcom Software
MANITOBA	
The Pas	Jodi's Sight & Sound
Winnipeg	J & J Electronics Ltd.
NEW BRUNSWICK	
Moncton	Canadian Colour Connection
NEWFOUNDLAND	
Labrador City	Sound & Vision
NOVA SCOTIA	
Dartmouth	Sector Software
Halifax	Atlantic News
ONTARIO	
Angus	Micro Computer Service
Atikokam	Gill's Furniture
Bowmanville	Bowmanville Audio Vision LTD
Cochrane	Frederickhouse Sound
Etobicoke	NEPCOM
Hamilton	Galls Book World
Hanover	CMUG
Kingston	T.M. Computers
Ottawa	National News Co., Ltd.
South River	Max TV
Strathroy	Downtown Sound
QUEBEC	
LaSalle	Messageries de Presse Benjamin Enr.
Sherbrooke	Soc. De Loc
SASKATCHEWAN	
Assiniboia	Telstar News
Nipiwan	Cornerstone Sound
Regina	George Glass
	Regina CoCo Club
	Softwre Supermarket
Saskatoon	Computertime
Tisdale	Paul's Service
YUKON	
Whitehorse	Big Byte Computer Services
	H & O Holdings
PUERTO RICO	
San Juan	Software City

Also available at all Waldenbooks, Coles and selected B. Dalton Bookseller stores in the United States and Canada.

ADVERTISER'S INDEX

We encourage you to patronize our advertisers — all of whom support the TRS-80 Color and TDP System-100 computers. We will appreciate your mentioning **THE RAINBOW** when you contact these firms.

Heat Up Your COCO
With J&M'S *Hot* Disk Controller

DRIVE 0 SYSTEM $289
Upgrade your COCO by adding J&M's famous disk controller, our advanced JDOS operating system, and a top quality drive all for only $289.

Drive 0 System with one single side drive $289

Drive 0,1 System with two single side drives ... $429

Drive 0,1 System with two double side drives ... $499

JFD-COCO DISK CONTROLLER $139
The J&M Systems' JFD-COCO Disk Controller has set new standards in performance and quality. Gold contacts assure reliability, built-in digital phase lock loop data separator means NO adjustments, and the JFD-COCO is plug compatible with both the original COCO and the new COCO-2.

JDOS
JDOS implements all RS DOS basic commands, plus many more, including auto line numbering, up and down arrow keys for scrolling, DOS to boot OS/9*, FLEX*, and error trapping. JDOS supports RS compatible disk formats, plus handles 40 track single side and double side drives.

DISK DRIVES
A drive is just a drive without a case. We manufacture our own high quality cases and TransPower power supplies. Gold contacts are brought out at the back for easy connection.

MEMORY MINDER *
Memory Minder is a disk drive test program that makes the following major checks of your drives without disassembly or special test equipment: head alignment, disk speed, index hole timing, azimuth, hysteresis, read sensitivity, and clamping. Memory Minder can be used to actually align the drives while viewing the graphics on the screen. This program is a must for anyone who values the data that is saved on diskettes.

MM-COCO-1 Memory Minder for single side drives.................................. $79

MM-COCO-2 Memory Minder for double side drives.................................. $99

* FLEX is a registered trademark of Technical Systems Consultants, Inc.
* OS/9 is a registered trademark of Microware, Inc.
* Memory Minder is a registered trademark of J&M Systems, Inc.

To order, call (505) 292-4182, or send payment with order to:

J & M SYSTEMS, LTD.
15100 CENTRAL SE
ALBUQUERQUE, NEW MEXICO 87123
505/292-4182

We accept MasterCard and Visa